Adaptive infrastructure. On-demand.

CLI Storage-Management Guide

Part Number: 810-0044-00, Revision G

Acopia Networks®, Inc.
41 Wellman Street
Lowell, MA 01851
(978) 513-2900 tel
(978) 513-2990 fax

CLI Storage-Management Guide

Revision History

September 2006 - Rev A
October 2006 - Rev B, updates for Software Release 2.4.2
January 2007 - Rev C, updates for Software Release 2.4.3
March 2007 - Rev D, updates for Software Release 2.5.0
May 2007 - Rev E, updates for Software Release 2.5.1
August 2007 - Rev F, add forest-to-forest trusts for Software Release 2.6.0
November 2007 - Rev G, minor changes in "show" commands, Software Release 2.7.0

Contents

Chapter 9

Adding a Managed Volume

Chapter 12

Policy for Balancing Capacity

Chapter 15

Shadowing a Volume

Chapter 1

Introduction

This manual contains instructions and best practices for setting up and managing storage on the Adaptive Resource Switch (ARX®). These instructions focus on the Command-Line Interface (CLI).

Use this book after the ARX is installed and connected to its clients and servers through IP. The platform's *Hardware Installation* manual explains how to install the ARX. You can set up basic networking through a GUI wizard (as described in the *GUI Quick Start: Network Setup*), or work with the more-advanced CLI features described in the *CLI Network-Management Guide*.

The ARX

The Adaptive Resource Switch (ARX®) is a highly available and scalable solution that brings resource awareness to a file storage infrastructure, and adapts these resources to meet the demands of users and applications in real time. The ARX provides a file-virtualization layer that aggregates the total capacity and performance of your file storage. A *namespace* provides location-independent, transparent mapping of user requests onto the appropriate storage resource. You can configure policies that the switch enforces for the placement, replication and migration of files. Through policy configuration, the ARX adapts to the real-time demands of users and applications. The ARX thereby serves as a *resource proxy* for the files and services behind it.

Back-end Storage and Servers

The Adaptive Resource Switch aggregates heterogeneous file systems and storage into a unified pool of file storage resources. Through this unification, you can manage these resources to adapt to user demands and client applications. File storage assets can be differentiated based on user-defined attributes, enabling a class-of-storage model. You can reclaim stranded capacity through policy implementation for more effective storage utilization, and you can add capacity without disruption. Back-end resources are monitored for availability and performance, as well as user-access patterns that drive policy decisions.

Front-end Services

The Adaptive Resource Switch acts as an in-band file proxy for the Network File System (NFS) and Microsoft's Common Internet File System (CIFS) protocols. *Front-end services* provide the file virtualization layer that masks the physical file storage from the user and application. The switch becomes the file access point, as opposed to the actual physical resource, providing file access through a namespace. Users and applications maintain a single consistent file path that is transparently mapped to the proper physical resource where the information resides.

Policy

The Adaptive Resource Switch provides policy-based resource switching. Through *policy* configuration, you can optimize the placement of files onto the appropriate storage resources and automatically adapt these resources based on user and application demand. The ARX performs file replication and migration based on performance, usage or other life-cycle characteristics, enabling you to implement a flexible file services strategy. Examples of policies include: migrating files to reclaim stranded capacity; migrating files across different tiers of storage based on access patterns and/or value; and replicating frequently accessed files for performance. The result is more efficient utilization and greater flexibility in file storage management.

Resilient Overlay Network (RON)

You can connect multiple ARXes with a Resilient Overlay Network (RON), which can reside on top of any IP network. This provides a network for distributing and accessing file storage. ARXes can replicate storage to other switches in the same RON, updating the replicas periodically as the writable master files change. This is called a *shadow copy*, where a source volume on one switch periodically copies its files to one or more *shadow volumes* on other switches. Clients can access the shadow volumes at multiple geographic locations, independent to where the source volume resides.

Audience for this Manual

This manual is intended for

- network technicians responsible for layer 1 and 2 networks,
- network engineers responsible for the Internet Protocol (IP) layer (layer 3),
- storage engineers who design and manage storage systems (SANs, NASes, and DASes), and
- crypto officers who manage all of the Critical Security Parameters (CSPs) of a network.

The text presumes that all readers are comfortable with a command-line interface (CLI), especially one based on the Cisco IOS.

Using this Manual

The next chapter shows some of the problems in today’s storage networks and provides high-level instructions for using the ARX to solve those problems.

The remaining chapters are presented in the same order that you would use to configure storage on a new ARX. Before you begin, you must follow the instructions in your *Hardware Installation Guide* to install the switch, set up its management IP, and prepare it for CLI provisioning. A network engineer can then use the *GUI Quick Start: Network Setup* manual or the *CLI Network-Management Guide* to set up the required networking parameters. Then you can follow the order of the chapters in this manual to

1. (for CIFS installations) configure some Windows security parameters referenced in other parts of the configuration,
2. (for NFS installations) add NIS netgroups and/or NFS access lists,
3. examine back-end filers (external NAS devices, or file servers with DAS) to determine their eligibility for use in a namespace,
4. add one or more back-end filers,
5. aggregate the filer storage into one or namespace volumes,
6. configure a global server and virtual server,
7. configure front-end services, such as NFS and CIFS, that clients can use to access the volume(s) through the global/virtual server,
8. configure namespace policy (capacity balancing, fileset configuration, and fileset-placement policy), and
9. configure a read-only shadow volume for a valued managed volume.

You can later return to any chapter to update the configuration at a particular layer.

Document Conventions

This manual uses the following conventions:

`this font` represents screen input and output;

- **`bold text`** represents input, and
- *`italic text`* appears for variable input or output.

this font is used for command-syntax definitions, which use the same rules for bold and italic. Command-syntax definitions also use the following symbols:

- [*optional-argument*] - square brackets ([ ]) surround optional arguments;

- *choice1* | *choice2* - the vertical bar (|) separates argument choices;
- {*choice1* | *choice2* | *choice3*} - curly braces ({ }) surround a required choice;
- [*choice1* | *choice2*]* - an asterisk (*) means that you can choose none of them, or as many as desired (for example, "choice1 choice2" chooses both);
- {*choice1* | *choice2*}+ - a plus sign (+) means that you must choose one or more.

CLI Overview

The Command-Line Interface (CLI) has its commands grouped into modes. Modes are structured as a tree with a single root, *exec* mode. This section summarizes the mode structure and explains some CLI conventions.

Exec Mode

When you log into the CLI, you begin in exec mode. If the hostname is "bstnA6k," the prompt appears as shown below:

```
bstnA6k>
```

You can access all global commands (such as show commands) from exec mode, and you can use the enable command to enter priv-exec mode.

```
bstnA6k> enable
```

Global Commands

You can access global commands from any mode, not just exec. Global commands include all show commands, terminal commands, commands for working with local maintenance files (such as log files), and commands for running CLI scripts.

Priv-exec Mode

Priv-exec mode has the following prompt:

```
bstnA6k#
```

Priv-exec mode contains chassis-management commands, clock commands, and other commands that require privileges but do not change the network or storage configuration.

Priv-exec has two sub modes, cfg and gbl.

Cfg Mode

To enter cfg mode, use the config command:

```
bstnA6k# config
bstnA6k(cfg)#
```

Config mode contains all modes and commands for changing the configuration of the local switch, such as network configuration.

Gbl Mode

To enter gbl mode, use the global command:

```
bstnA6k# global
bstnA6k(gbl)#
```

Gbl mode controls all parameters that are shared in an HA pair, such as namespaces and global servers.

Exiting a Mode

From any mode, use the exit command to return to its parent mode. From priv-exec mode, this command exits the CLI; to go from priv-exec mode back to exec mode, use the no enable command.

From any submode of cfg or gbl mode, you can return immediately to priv-exec mode by using the end command or pressing <Ctrl-z>.

Prompts

Prompts contain information about your position in the mode hierarchy as well as the name of the object you are configuring. For example, suppose you use the following command in gbl mode:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])#
```

This command places you into a new mode, as indicated by the new CLI prompt. The prompt shows the name of the mode, “gbl-ns,” and the name of the configuration object, a namespace called “wwmed.” Abbreviations are used for mode names (for example, “ns” instead of “namespace”) to conserve space on the command line.

When you descend to lower modes in the config tree, the prompt offers more information. To extend the previous example, suppose you enter the following command to configure the “/local” volume in the wwmed namespace:

```
bstnA6k(gbl-ns[wwmed])# volume /local
bstnA6k(gbl-ns-vol[wwmed~/local])#
```

The tilde character (~) separates a parent object from its child: “wwmed~/local” shows that you are in the “/local” volume under the “wwmed” namespace.

The no Convention

Most config commands have the option to use the “no” keyword to negate the command. For commands that create an object, the no form removes the object. For commands that change a default setting, the no form reverts back to the default. As an example,

```
bstnA6k(gbl-ns[wwmed])# no volume /local
```

removes the “/local” volume from the “wwmed” namespace.

The enable/no enable Convention

Many objects and configurations require you to enable them using the enable command before they can take effect. Likewise, many objects and configurations require you to first disable them using the no enable command before you can complete a related command or function. The no enable command does not remove an object; it only disables it until you re-enable it. The enable/no enable commands exist in many modes and submodes in the CLI.

For example, the following command sequence enables the namespace named “wwmed:”

```
bstnA6k(gbl)# namespace wwmed
```

```
bstnA6k(gbl-ns[wwmed])# enable
bstnA6k(gbl-ns[wwmed])# ...
```

Getting Started

For the initial login, refer to the instructions for booting and configuring the switch in the appropriate *Hardware Installation Guide*.

For subsequent logins, use the following steps to log into the Acopia CLI:

1. If you are on-site, you can connect a serial line to the serial console port. This port is labeled ‘Console;’ you can find it on the front panel of the switch. By default, the port is set for 9600 baud, 8, N, 1.

   You can also telnet to the switch’s management interface. For example:

   telnet 10.10.10.10

   In either case, a login prompt appears:

```
Username:
```

2. Enter your username and password. For example:

```
Username: admin
Password: acopia
```

   The CLI prompt appears:

```
SWITCH>
```

   The name, “SWITCH,” is the default hostname. The hostname is reset as part of the initial-boot process, so it is likely that yours will differ.

Entering Cfg or Gbl Mode

The CLI contains two major configuration modes: cfg and gbl. The cfg mode contains submodes for configuring locally-scoped parameters, only applicable to the local ARX. These parameters include layer-2, layer-3, and chassis configuration. Gbl mode applies to configuration that is shared among both switches in a redundant pair, such as namespaces and global servers.

After you log into the CLI, use the config command to enter cfg mode:

```
SWITCH> enable
SWITCH# configure
SWITCH(cfg)#
```

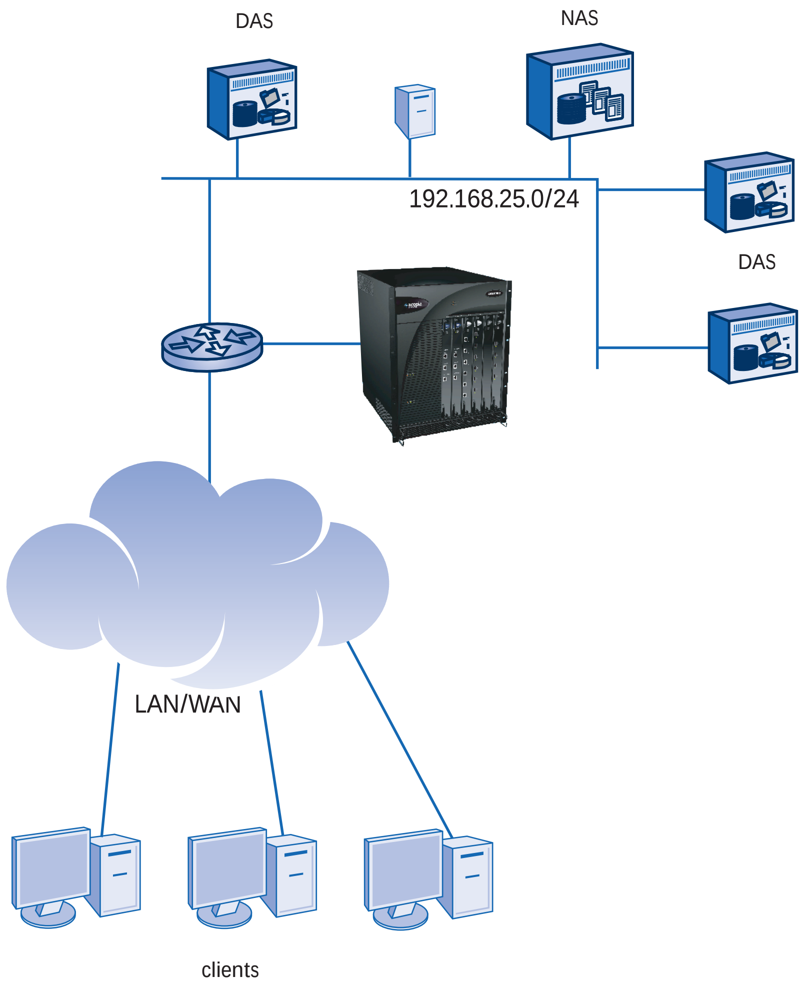
To enter gbl mode, use the global command instead:

```
SWITCH> enable
SWITCH# global
SWITCH(gbl)#
```

The command sequences in this manual all begin either in cfg mode or gbl mode.

Sample Network

The examples in this manual draw from a single, fictitious network. The network filers all live on a class-C subnet at 192.168.25.x. These filers are called *back-end* filers, since they are the storage behind the *front-end* services of the ARX. The filers can be heterogeneous: NAS devices and file servers (possibly with additional DAS) need only support CIFS or NFS to be on the back end of the ARX.



Contacting Customer Service

You can use the following methods to contact Acopia Customer Service:

E-mail	support@acopia.com
Telephone	1-866-4Acopia (1-866-422-6742)
Acopia TAC Online Acopia's online customer knowledge base and support request system	http://www.acopia.com/support/

Chapter 2

Product Overview

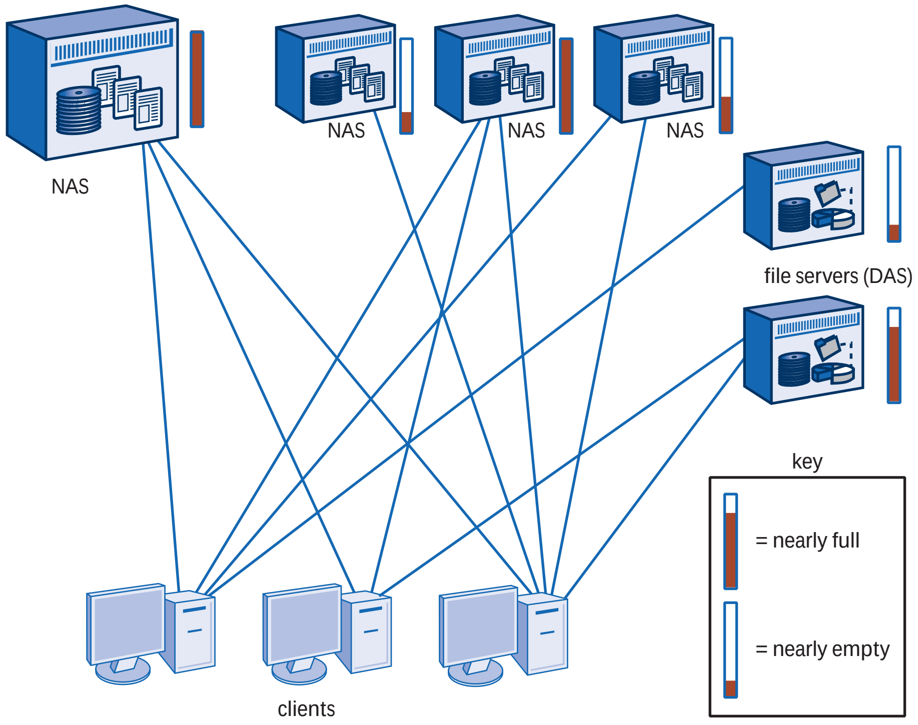
Solutions to Common Storage Problems

This chapter shows some of the problems inherent with today's file-storage networks, then it demonstrates the solutions offered by the ARX. References to relevant chapters appear at the end of each solution, so that you can configure the solutions in your network.

Today's File Storage

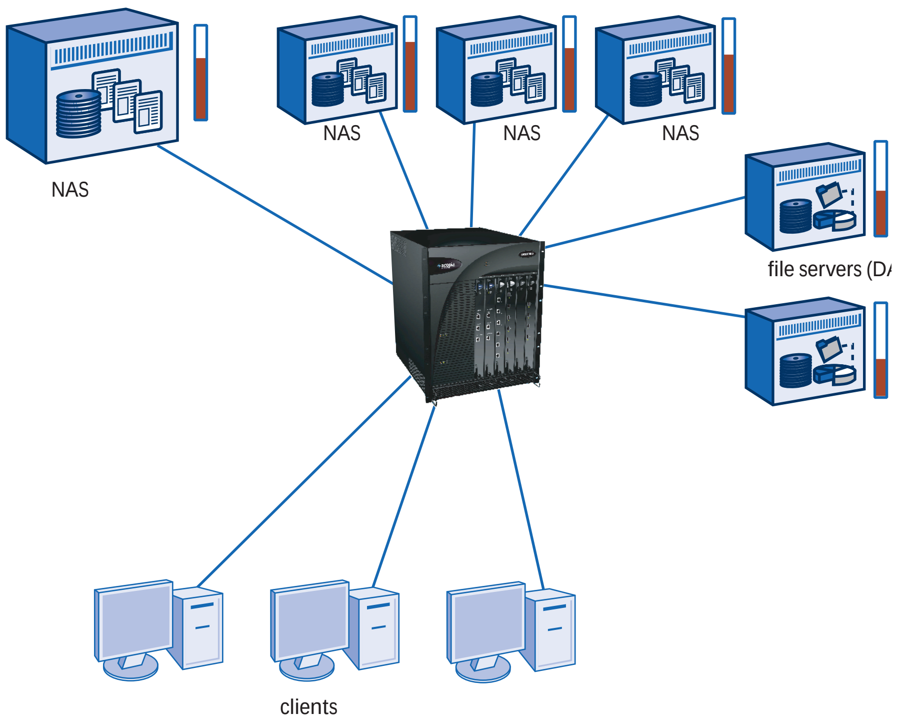
Today's storage networks typically evolve over time into storage *islands* with imbalanced capacities. As you add new storage devices to the network, clients view each device as a discrete set of CIFS shares or NFS exports. Inevitably, some shares are more popular than others, causing load imbalance amongst the storage devices. Storage engineers try to match popular shares with storage devices that can handle the most load, but client preferences change and client writes are unpredictable.

Balancing the capacity between these islands means moving popular files between file servers, but this can be difficult. Each client connects to back-end storage statically, through an IP address or FQDN, and chooses from a list of shares and paths that reside at that storage device. Moving files from one storage device to another means updating the client-side view of IP addresses, share names, and/or file paths. File storage is therefore static and expensive to manage.



Adaptive Resource Switching solves these expensive problems by creating a virtual view of back-end storage for the front-end clients. Clients connect to all back-end storage through a virtual directory structure called a *namespace volume*. The storage devices are hidden behind the volume, making it possible to balance capacity and load at the back end without affecting any client-access on the front-end. An Adaptive Resource Switch can thereby

- *optimize* the file-storage infrastructure,
- *adapt* file storage to client demands, and
- *control* management costs.

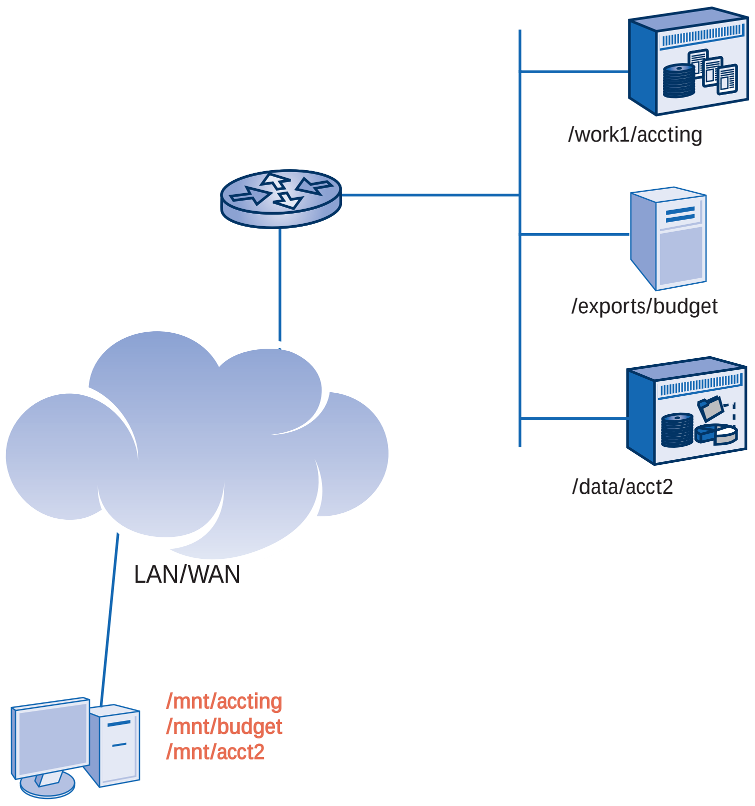


Acopia's Adaptive Resource Switch can optimize, adapt, and control your storage resources through namespace configuration and file migration. The sections below summarize the configuration steps for each of these solutions.

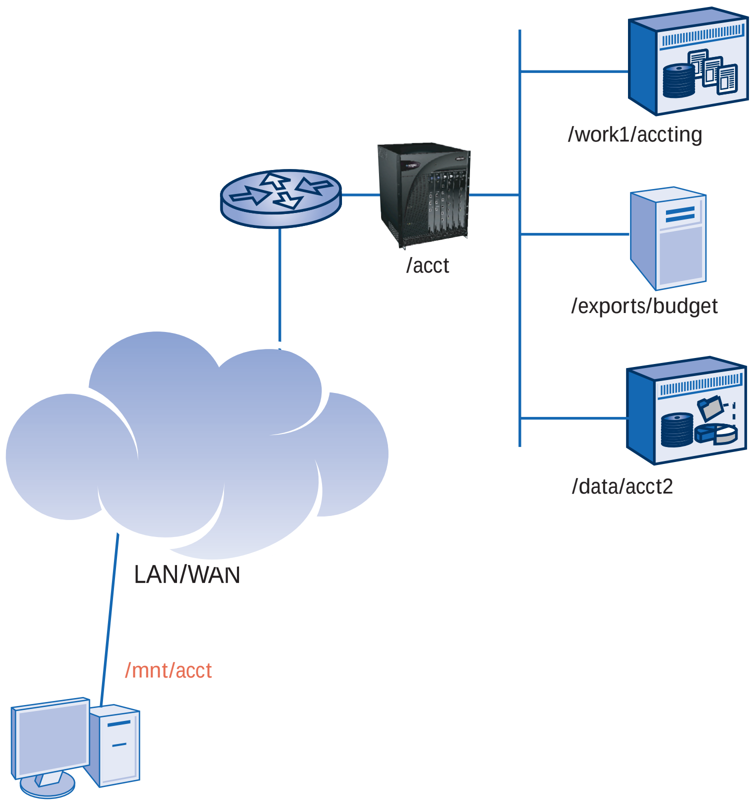
Optimizing Storage in a Namespace Volume

A *namespace* is a group of file systems under a single authentication domain. A namespace is comprised of one or more *volumes*, where each volume is like a discrete file system. A volume is composed of *shares*, where each share maps to an export or share on an actual back-end filer. The volume can contain shares from multiple back-end filers, but the client sees a single mount-point or share-point.

Consider three filers with one NFS export each. The figure below shows the filers behind a standard router. In this configuration, the client must issue three mounts to access all three exports.



The ARX can aggregate all three exports into a single namespace volume, "/acct" in this example. The client then only needs to mount the single, aggregated volume.



The client now connects to the ARX rather than the individual filers. This creates an opportunity for upgrading storage on the back-end without changing the front-end view.

Through storage aggregation in a namespace volume, the ARX simplifies the client's interface to multiple storage devices and creates new opportunities for storage maintenance.

Configuration Instructions

Namespace configuration constitutes the bulk of switch configuration. You begin by connecting to back-end filers and aggregating their storage.

1. Chapter 6, *Adding an External Filer*, contains instructions for adding an *external* (NAS or DAS-enhanced) filer to the configuration.
2. Chapter 7, *Configuring a Namespace*, contains instructions for aggregating external-filer storage into a namespace.

Once the namespace is configured, you must configure the server for clients to access the namespace:

1. configure a *global server* that clients can use to access the namespace over IP (see Chapter 10, *Configuring a Global Server*), and
2. create one or more front-end services, such as NFS or CIFS, to offer the namespace storage through the global server. See Chapter 11, *Configuring Front-End Services*.

Adapting Storage to User Demands

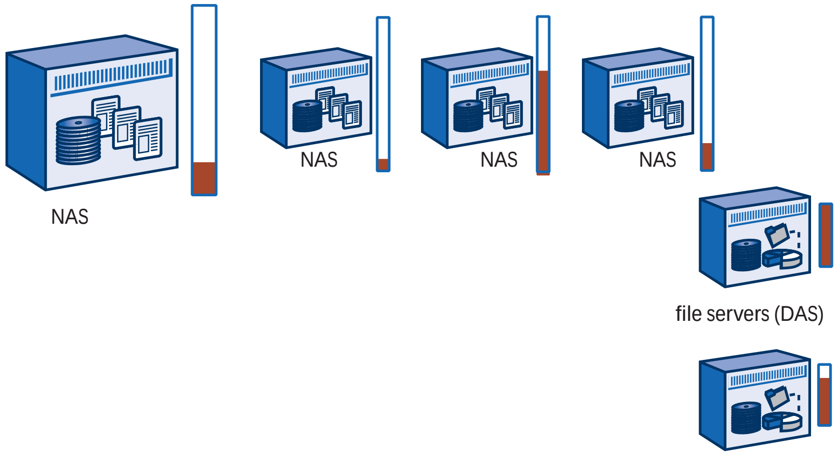
The ARX's position between clients and back-end servers presents a unique opportunity for adapting to changes in usage. The ARX processes every client request and server response, so it can

- *classify* information as it is generated,
- *monitor* client-access patterns and back-end-resource usage, and
- *adapt* the back-end resources to meet client demands in real time.

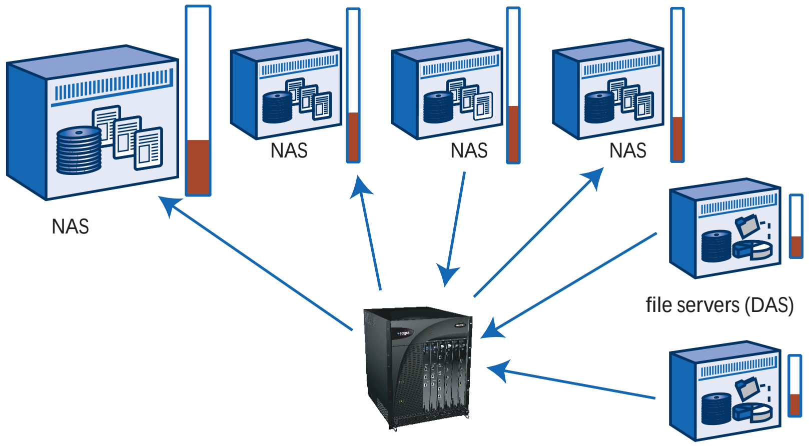
The ARX adapts by migrating files from one filer to another. You use namespace policy to set rules and thresholds for migration.

Migration for Capacity

The ARX can keep all filers at or above the same minimum free space, so that overburdened filers can offload their files to other filers. This is called *auto-migration* off of the filer that is low on free space. You configure this by declaring a share farm and establishing the auto migration rule in the share farm. The internal policy engine balances capacity by migrating popular files from an over-filled share to the shares with more free space. For example, consider a scenario where several larger filers are under-utilized and two DAS-enhanced file servers are nearly filled to capacity:



An auto-migrate rule migrates files off of the over-burdened filers and onto the filers with more available free space. In addition, the ARX ensures that no filer is over-filled; all filers in the share farm maintain the minimum free space until/unless they all fill up to this level.



Configuration Instructions

An auto-migrate rule is one of the share-farm rules described in Chapter 12, *Policy for Balancing Capacity*. See "Auto Migrating Existing Files" on page 12-18.

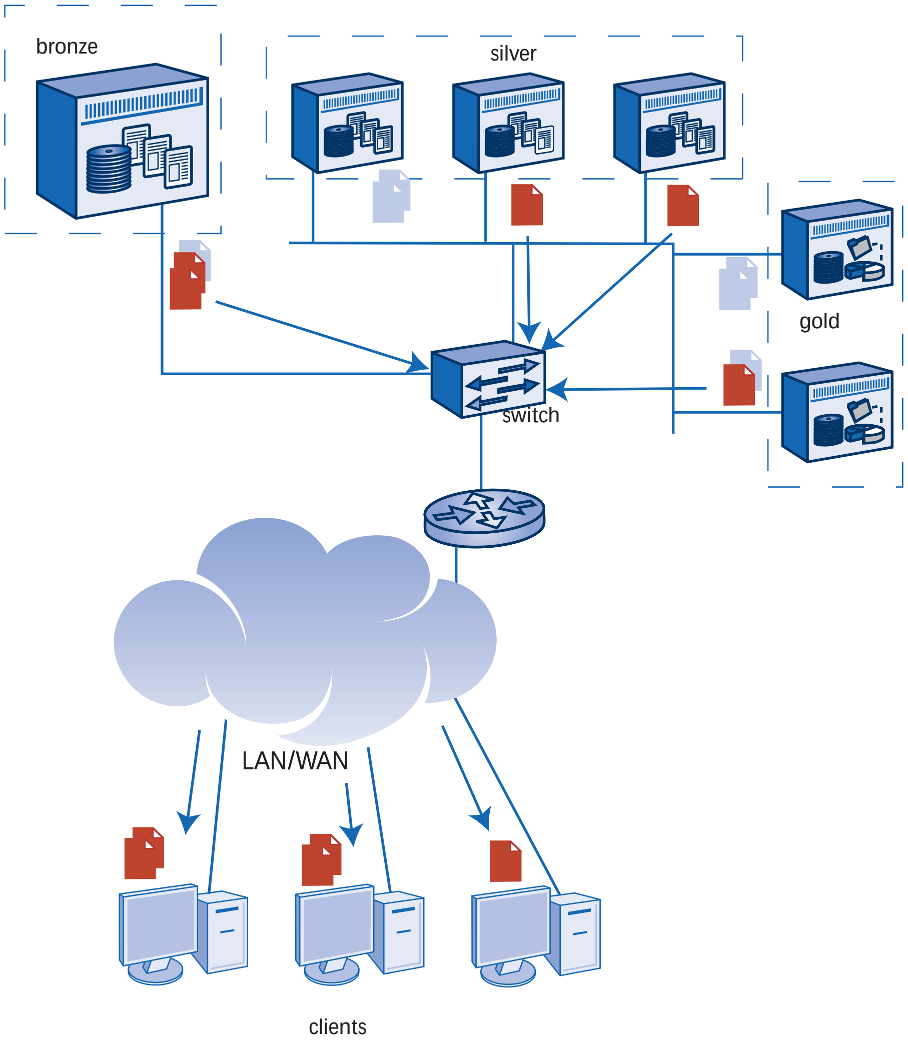
Migration for Class of Storage: File-Placement Policy

Another application for namespace policy is adaptive migration of files onto selected back-end storage. This migration is configured as a *file-placement* policy. To configure file-placement policy, you can

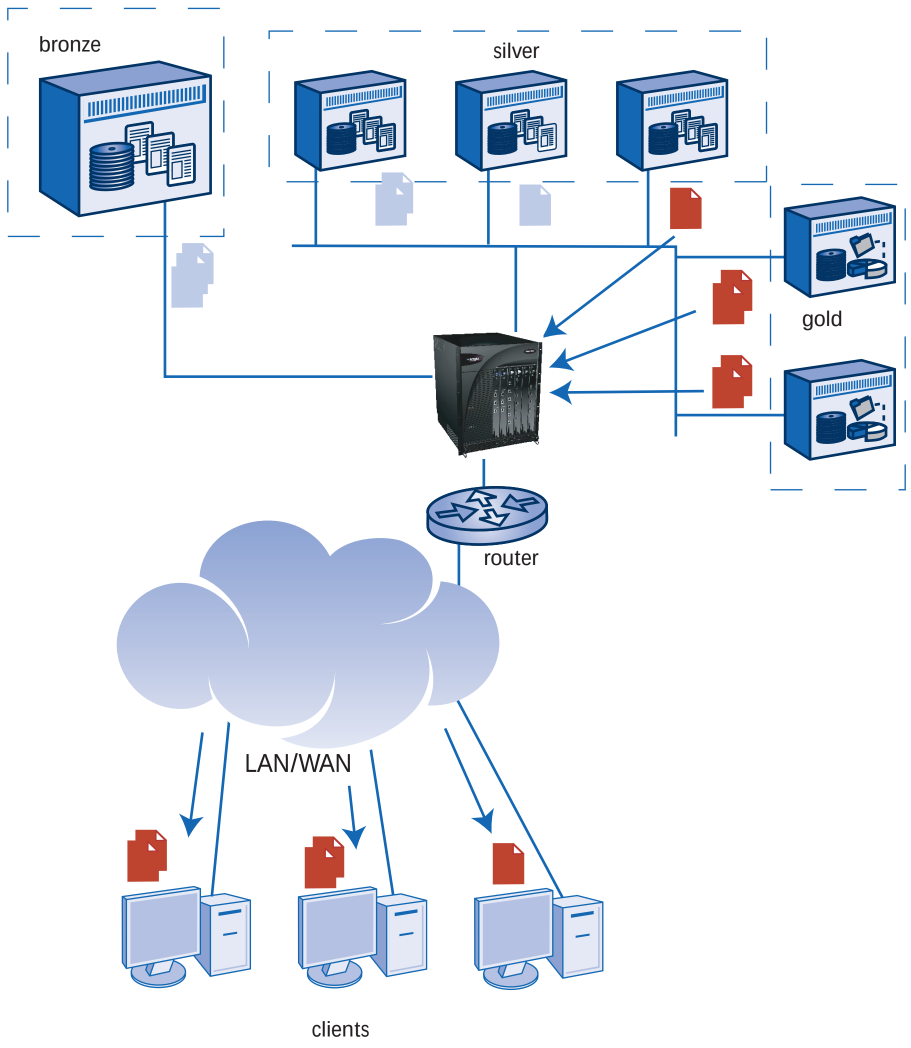
1. group files into *filesets* based on criteria like names (such as *.xml or *.mpg), location (/usr/local), and/or last-modified times,
2. differentiate your back-end filers by classifying them into different *share farms* (optional), and
3. configure a file-placement policy to place a fileset onto a given share or share farm.

The policy engine periodically re-examines its back-end files to re-group them into filesets, then it moves them to the configured back-end share or share farm.

Consider a site with several tiers of storage: a gold tier of expensive file servers, a silver tier of more-plentiful (perhaps slower) filers, and a bronze tier of least-expensive filers. Initially, administrators distribute files among their filers based on best guesses at the usage of the various files. Periodically, files in the bronze tier become unexpectedly "hot," impacting performance on that tier of the storage network.



File-placement policy can solve this problem. You can configure an age-based fileset that groups all files in the namespace based on their last-accessed times. This fileset could divide the files into weekly groups: files accessed this week, two-to-four weeks ago, and any time before four weeks ago. A file placement policy could then place the files on the correct tiers based on when they were last accessed. The ARX dynamically ensures that the most-popular files reside on the storage best-equipped to serve them.



Configuration Instructions

To configure a fileset, see Chapter 13, *Grouping Files into Filesets*.

To group your namespace shares into a share farm, see "Adding a Share Farm" on page 12-15.

For instructions on moving the fileset to your chosen storage, see Chapter 14, *Migrating Filesets*.

Controlling Costs

The ARX controls costs by optimizing network resources behind a namespace and adapting to client usage in real time. By configuring a namespace and namespace policies, you can

- reclaim existing storage resources (with share-farm balancing),
- expand your storage-purchasing options (by grouping filers into share farms), and
- simplify file-storage management (by hiding filers behind a namespace volume).

Chapter 3

Preparing for CIFS Authentication

The ARX is a file proxy between clients and back-end filers; it must authenticate clients on the front end, and it must provide valid credentials to servers on the back end. To set up the switch proxy in an CIFS environment, you must configure two sets of authentication parameters in advance:

- A *proxy user*, configured with a valid username and password.

  The switch uses the proxy-user credentials when it performs autonomous operations (such as moving files between shares).

- (NTLM only) A mechanism to facilitate client authentication. This is typically an external *NTLM-Authentication server*, the host machine for the separately-installed NTLM *Secure Agent* software.

- (Kerberos only) The structure of the *Active-Directory forest* in your network. This identifies the Domain Controllers (DCs) for each Windows domain, and it provides the ARX with the Windows-domain hierarchy.

You must configure these Windows-security parameters first so that you can reference them later.

You can also define the permissions for a group of CIFS clients to use Windows-Management applications, such as the MicroSoft Management Console (MMC), in CIFS namespaces. You can create one or many such management-authorization groups. You can later assign each group to one or more CIFS namespaces.

Concepts and Terminology

A *namespace* is an aggregated view of several back-end filers. Each namespace operates under a single authentication domain, the same domain supported by all of its back-end filers. This applies to both Windows and Unix domains.

A *global server* is an client-entry point to the services of the ARX. A global server has a fully-qualified domain name (such as myserver.mycompany.com) where clients can access namespace storage.

Adding a Proxy User

Before you configure a namespace with Windows NTLM or Kerberos, you must configure a proxy user for the namespace. A proxy user is a single username/password in a particular Windows domain. The ARX uses the proxy-user as its identity while reading from CIFS shares (to import the share into a namespace) and moving files between shares (for capacity balancing and other policies).

Note

A proxy user must belong to the Backup Operator’s group, to ensure that it has sufficient authority to move files freely from share to share.

From gbl mode, use the proxy-user command to create one proxy user:

proxy-user *name*

where ***name*** (1-32 characters) is a name you choose.

This puts you into gbl-proxy-user mode, where you set the Windows domain, username, and password for the proxy user. When you later configure a namespace in the same Windows domain, you can apply this proxy-user configuration to that namespace. You can apply the same proxy user to multiple namespaces in the same domain.

For example, the following command sequence creates a proxy user named “acoProxy2:”

```
bstnA6k(gbl)# proxy-user acoProxy2
```

```
bstnA6k(gbl-proxy-user[acoProxy2])# ...
```

Specifying the Windows Domain

The first step in configuring a proxy user is to specify its Windows domain. From gbl-proxy-user mode, use the windows-domain command to specify the domain:

windows-domain *name*

where ***name*** is 1-64 characters.

For example, the following command sequence specifies the "MEDARCH" domain for the "acoProxy2" proxy user:

```
bstnA6k(gbl)# proxy-user acoProxy2
bstnA6k(gbl-proxy-user[acoProxy2])# windows-domain MEDARCH
bstnA6k(gbl-proxy-user[acoProxy2])# ...
```

Removing the Windows Domain

Removing the domain from the proxy-user configuration effectively disables the proxy user. From gbl-proxy-user mode, use no windows-domain to remove the domain:

no windows-domain

For example:

```
bstnA6k(gbl)# proxy-user testuser
bstnA6k(gbl-proxy-user[testuser])# no windows-domain
bstnA6k(gbl-proxy-user[testuser])# ...
```

Specifying the Username and Password

The final step in configuring a proxy user is to specify a username and password. This username/password must belong to the Backup Operator's group to ensure that it has sufficient authority to move files freely from share to share.

Note

This applies to all back-end filers and shares; a user can be defined as part of the Backup Operator's group on one CIFS filer but not on another.

From gbl-proxy-user mode, use the user command to specify the username:

user *username*

where ***username*** (1-64 characters) is a valid username in the proxy-user's domain.

The CLI prompts you for the user's password, then prompts to validate the password.

For example, the following command sequence specifies the username, "jqpublic:"

```
bstnA6k(gbl-proxy-user[acoProxy2])# user jqpublic
Password: jqpasswd
Validate Password: jqpasswd
bstnA6k(gbl-proxy-user[acoProxy2])# ...
```

Removing the Username and Password

Removing the username from the proxy-user configuration effectively disables the proxy user. From gbl-proxy-user mode, use no user to remove the username:

no user

For example:

```
bstnA6k(gbl)# proxy-user testuser
bstnA6k(gbl-proxy-user[testuser])# no user
bstnA6k(gbl-proxy-user[testuser])# ...
```

Listing All Proxy Users

You can use the show proxy-user command to get a list of all proxy users on the ARX:

show proxy-user

For example:

```
bstnA6k(gbl)# show proxy-user

Name                               Domain           User
--------------------------------------------------------------------------------
acoProxy1                          WWMEDNET         jqprivate
acoProxy3                          FDTESTNET        jqtester
acoProxy2                          MEDARCH          jqpublic
bstnA6k(gbl)#
```

Showing One Proxy User

To focus on one proxy user, you can specify a name in the show proxy-user command:

show proxy-user *name*

where ***name*** (1-32 characters) identifies the proxy user to show.

For example:

```
bstnA6k(gbl)# show proxy-user acoProxy2

Name                               Domain           User
--------------------------------------------------------------------------------
acoProxy2                          MEDARCH          jqpublic
bstnA6k(gbl)#
```

Removing a Proxy User

From gbl mode, use no proxy-user to remove a proxy-user configuration:

no proxy-user *name*

where ***name*** (1-32 characters) identifies the proxy user to remove.

For example, the following command sequence removes a proxy user called proxyNYC:

```
bstnA6k(gbl)# no proxy-user proxyNYC
bstnA6k(gbl)# ...
```

Configuring the NTLM Authentication Server

Before you configure a namespace with Windows NTLM, you must also configure an NTLM-authentication server for the namespace. The *NTLM Authentication Server* is the Windows Domain Controller (DC) that is the host for the Acopia *Secure Agent* software. When the ARX gets a request for access from a CIFS client, it passes the password to the Secure Agent for authentication. As a CIFS proxy, the ARX must also access back-end CIFS filers; the Secure Agent answers all password challenges as the same client.

The Secure Agent is required because the ARX acts both as the CIFS server for an end user *and then* as a CIFS client on behalf of the same user. The user may request access to multiple back-end filers, so the ARX must answer multiple NTLM challenges on behalf of the client. The NTLM protocol prevents the ARX from holding onto the user's password, so a secure mechanism is required for retrieving the user's password as needed. The Secure Agent, residing on the Windows Domain Controller, provides this mechanism. The ARX passes the NTLM authentication challenge and the client's username to the Secure Agent, which retrieves the client's password to answer the NTLM challenge. (The Secure Agent accesses the DC's SAM database to get the client's password hash.) The ARX forwards the challenge response to the back-end filer, completing the authentication session.

You separately install the Acopia Secure Agent at a DC, then you specify the server's IP address (and other parameters) at the ARX's CLI. Refer to the *Secure Agent Installation Guide* for instructions to install the Secure Agent and then configure it at the ARX.

Listing NTLM Authentication Servers

Use the show ntlm-auth-server command to display summary information on one or more configured Secure Agent servers. This command shows where the servers are located in the local database.

show ntlm-auth-server [*name*]

where ***name*** (optional, 1-128 characters) is the name of a Secure Agent server instance. If no name is specified, this command shows names for all configured Secure Agent servers.

For example, the following command lists the only configured Secure Agent server:

```
bstnA6k> show ntlm-auth-server


Name
---------------------------------------
dc1


bstnA6k> ...
```

To continue the example, this command shows the details for the “dc1” server:

```
bstnA6k> show ntlm-auth-server dc1

Name                 Domain Name      Server           Port
--------------------------------------------------------------------------------
dc1                  MEDARCH          192.168.25.102   25805

Mapped to the Following Namespaces
--------------------------------------------------------------------------------
bstnA6k> ...
```

Displaying Detailed Server Status

Use the show ntlm-auth-server status command to display detailed status information on one or more configured Secure Agent servers. This command shows how the servers are configured to a switch.

show ntlm-auth-server status [*name*]

where ***name*** (optional, 1-128 characters) is the name of a Secure Agent server instance. If no name is specified, this command shows names for all configured Secure Agent servers.

For example, the following command displays detailed status information on a Secure Agent server at Connection #1 and Connection #2.

```
bstnA6k> show ntlm-auth-server status

******************* SECURE AGENT STATISTICS ******************
Agent IP   : 192.168.25.102
Agent Port : 25805

Uptime: 287 days, 15 hours, 52 minutes and 4 seconds
Current Connections: 28
Failed connection attempts: 326
Successful connection attempts: 73120
Account Scan Interval: 300
ADSI status : disabled
Software version: Version 1.02.000.06275 (Nov  7 2004 22:23:52) [nbuilds]

Connection #1
  Source IP: 10.54.220.200
  Duration: 138335 (seconds)
  bytes Received: 48
  bytes Transmitted: 168
  Successful Client Authentications: 1
  Failed Client Authentications:
    No Such User     : 0
    Bad Password     : 0
    Locked out       : 0
    Account Disabled : 0
    Account Expired  : 0
    Password Expired : 0
    Time Restricted  : 0
    API Error        : 0
  Filer Response Generation:
    Success count: 1
    Failure count: 0

Connection #2
  Source IP: 10.61.101.200
  Duration: 56192 (seconds)
  bytes Received: 9576
  bytes Transmitted: 28768
  Successful Client Authentications: 397
  Failed Client Authentications:
    No Such User     : 0
    Bad Password     : 0
    Locked out       : 0
    Account Disabled : 0
    Account Expired  : 0
    Password Expired : 0
    Time Restricted  : 0
    API Error        : 0
  Filer Response Generation:
    Success count: 399
    Failure count: 0
...
```

Adding an Active-Directory Forest (Kerberos)

To prepare for a CIFS service that uses Kerberos to authenticate its clients, you must first create an *Active Directory forest*. This mimics the Active Directory (AD) forest in your Windows network. When a client accesses the CIFS front-end service from one of the domains in the AD forest, the switch uses this information to locate the appropriate DC.

You can skip this section if you are not using Kerberos with any CIFS service.

From gbl mode, use the active-directory-forest command to create a forest:

active-directory-forest *forest-name*

where ***forest-name*** (1-256 characters) is the name of the forest.

For example, the following command creates the 'medarcv' forest:

```
bstnA6k(gbl)# active-directory-forest medarcv
bstnA6k(gbl-forest[medarcv])# . . .
```

This places you into gbl-forest mode, where you create the various components of the AD forest. A forest consists of a *forest root* domain and one or more *child domains* in the same domain namespace (for example, "myisp.net" can be a forest root with two children, "myregion.myisp.net" and "yourregion.myisp.net"). Child domains can also be parents to more child domains, as dictated by their names ("myregion.myisp.net" can be the parent of "mylocale.myregion.myisp.net"). If there are trusted trees outside of this domain namespace (such as "telco.com"), you can add them as *tree domains*.

Identifying the Forest Root

The next step in creating an Active Directory forest is to identify the DC and domain for the forest root. From gbl-forest mode, use the forest-root command:

forest-root *domain-name ip-address*

where

domain-name (1-256 characters) identifies the AD domain of the forest root, and

ip-address is the IP address (for example, 10.120.95.56) of the forest root's DC.

For example, this command sequence selects the forest root for the 'medarcv' forest, 'MEDARCH.ORG:'

```
bstnA6k(gbl)# active-directory-forest medarcv
bstnA6k(gbl-forest[medarcv])# forest-root MEDARCH.ORG 192.168.25.102
bstnA6k(gbl-forest[medarcv])# . . .
```

Adding a Redundant Forest Root

Some networks configure redundant DCs to manage the AD-forest root. You can re-use the forest-root command to identify a redundant DC. The domain name is the same, but the IP address for the DC is different. For example, this command sequence configures redundant DCs for the 'medarcv' forest:

```
bstnA6k(gbl)# active-directory-forest medarcv
bstnA6k(gbl-forest[medarcv])# forest-root MEDARCH.ORG 192.168.25.102
bstnA6k(gbl-forest[medarcv])# forest-root MEDARCH.ORG 192.168.25.103
bstnA6k(gbl-forest[medarcv])# . . .
```

Removing a DC for the Forest Root

If there are redundant DCs for this forest root, you can always remove one of them. To remove a forest root with only a single DC, there can be no child domains or sub trees in the forest, nor can there be any trust relationships between this forest and any other forest. (Child domains and forest-to-forest trusts are explained in the subsections below.)

Use the no forest-root command to remove a DC for the forest root.

no forest-root *domain-name domain-controller*

where

domain-name (1-256 characters) identifies the AD domain of the forest root, and

domain-controller is the IP address of the DC to remove.

For example, this command sequence removes the second (redundant) DC from the ‘medarcv’ forest root:

```
bstnA6k(gbl)# active-directory-forest medarcv
bstnA6k(gbl-forest[medarcv])# no forest-root MEDARCH.ORG 192.168.25.103
bstnA6k(gbl-forest[medarcv])# . . .
```

Identifying a Dynamic-DNS Server

Many Active-Directory networks use dynamic DNS to map CIFS host names to IP addresses. Whenever an ARX’s front-end CIFS service changes its host name or IP address, the service sends the hostname-to-IP mapping to one or more dynamic-DNS servers in the AD forest. No manual changes to the DNS configuration are required.

Up-to-date DNS configuration is required for Kerberos, which uses FQDNs in its authentication tickets instead of IP addresses.

RFCs 1034 and 1035 define basic DNS, and RFC 3645 defines the Microsoft-specific authentication extensions for dynamic DNS. The ARX implementation supports all of these standards; it does not support any other dynamic-DNS RFCs.

To prepare for dynamic DNS, you identify the dynamic-DNS servers in this forest. Later chapters explain how to configure a front-end CIFS service to use these dynamic-DNS servers. To identify one dynamic-DNS server, called a *name server*, use the name-server command in gbl-forest mode:

name-server *domain-name ip-address*

where

domain-name (1-255 characters) identifies the AD domain, and

ip-address is the IP address of the domain's name server. This might be the same IP as for the domain's DC; dynamic DNS often runs on the same DC that supports the domain.

You can enter this command multiple times, once for each name server. If you enter multiple name servers for a given AD domain, a CIFS service in this domain will attempt to register with each of them in turn until it succeeds. It stops registering on the first success.

For example, this command sequence identifies three dynamic-DNS servers for the 'MEDARCH.ORG' domain. The first, 192.168.25.102, is also the DC for the forest root:

```
bstnA6k(gbl)# active-directory-forest medarcv
bstnA6k(gbl-forest[medarcv])# name-server MEDARCH.ORG 192.168.25.102
bstnA6k(gbl-forest[medarcv])# name-server MEDARCH.ORG 192.168.25.103
bstnA6k(gbl-forest[medarcv])# name-server MEDARCH.ORG 192.168.25.104
bstnA6k(gbl-forest[medarcv])# . . .
```

Removing a Name Server

If you remove the only name server for an Active-Directory domain, any changes to front-end CIFS services in that domain will require manual updates to DNS. That is, if a CIFS service is added to or removed from the ARX, an administrator must add or remove the corresponding "A" record from the external DNS server. As long as at least one name server is assigned to the domain, this maintenance penalty is not necessary. The DNS database must be correct for the CIFS service or Windows clients cannot authenticate with Kerberos.

To remove a name server from an AD domain, use the no name-server command:

no name-server *domain-name domain-controller*

where

domain-name (1-255 characters) identifies the AD domain, and

domain-controller is the IP address of the name server to remove.

For example, this command sequence removes the second (redundant) name server from the ‘MEDARCH.ORG’ domain. Recall from the previous example that this leaves two more name servers for the domain.

```
bstnA6k(gbl)# active-directory-forest medarcv
bstnA6k(gbl-forest[medarcv])# no name-server MEDARCH.ORG 192.168.25.103
bstnA6k(gbl-forest[medarcv])# . . .
```

Adding a Child Domain

A child domain is the child to any other domain in the forest: the forest root, another child domain, or a separate tree domain (explained below). Its domain name indicates its parentage: it must have a configured domain as its parent. From gbl-forest mode, use the child-domain command to add a child domain to a forest.

child-domain *domain-name domain-controller*

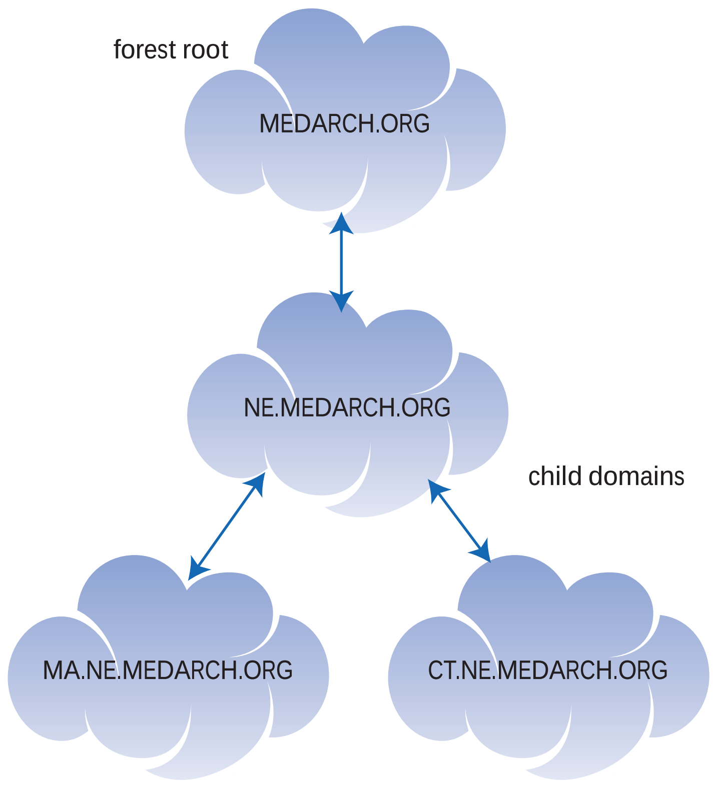
where

domain-name (1-256 characters) is the child domain name, and

domain-controller is the IP address (for example, 192.168.25.56) of the child domains’s DC.

As with the forest-root command, you can re-enter this command with the same ***domain-name*** to identify a redundant DC.

For example, this command sequence mimics the forest illustrated below:



The first child, "NE.MEDARCH.ORG," is a child of the root domain, "MEDARCH.ORG," and the last two domains are children under "NE.MEDARCH.ORG:"

```
bstnA6k(gbl)# active-directory-forest medarcv
bstnA6k(gbl-forest[medarcv])# child-domain NE.MEDARCH.ORG 172.16.124.73
bstnA6k(gbl-forest[medarcv])# child-domain MA.NE.MEDARCH.ORG 192.168.25.103
bstnA6k(gbl-forest[medarcv])# child-domain CT.NE.MEDARCH.ORG 10.10.167.40
bstnA6k(gbl-forest[medarcv])# . . .
```

Removing a Child Domain

The no child-domain command removes a child domain controller from a forest:

no child-domain *domain-name domain-controller*

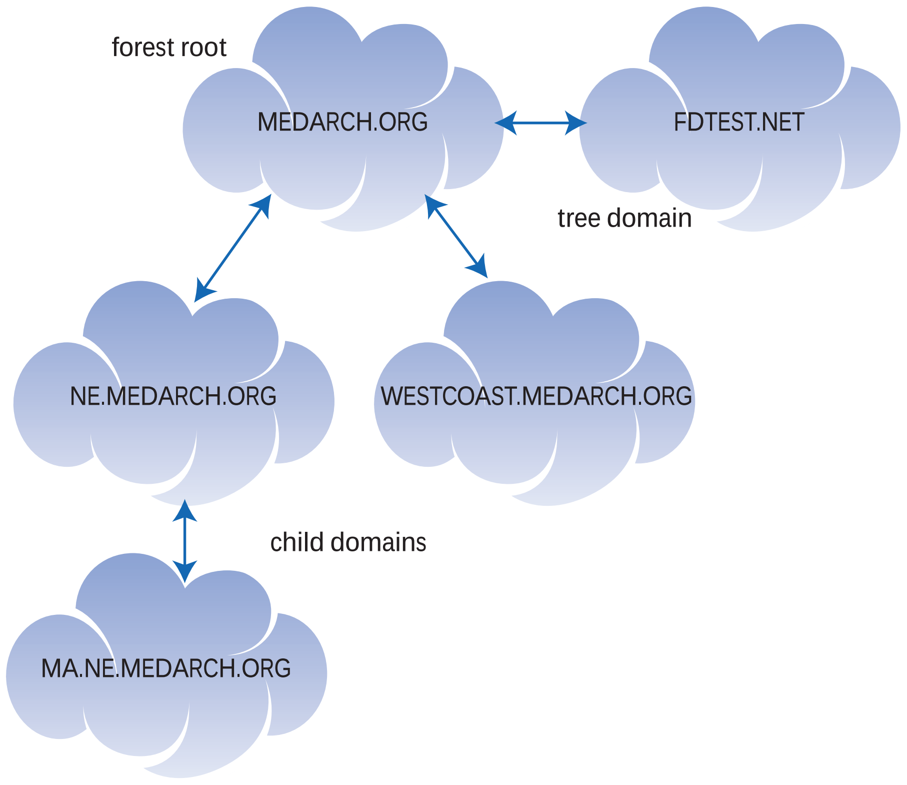
You can do this only if the child domain has a redundant DC, or if it has no children. Otherwise you must first add a redundant DC or remove all of the child domain's children.

For example, this command sequence removes a child domain with no children of its own:

```
bstnA6k(gbl)# active-directory-forest medarcv
bstnA6k(gbl-forest[medarcv])# no child-domain CT.NE.MEDARCH.ORG 10.10.167.40
bstnA6k(gbl-forest[medarcv])# . . .
```

Adding a Tree Domain

Some domains are outside the forest-domain namespace, but have two-way trust relationships with one or more of the forest's domains.



These are called *tree domains*. Use the tree-domain command to identify a tree domain and its DC:

tree-domain *domain-name domain-controller*

where

domain-name (1-256 characters) is the tree-domain name, and

domain-controller is the IP address (for example, 192.168.25.56) of the DC for the tree.

You can specify redundant DCs for a tree domain; enter the tree-domain command once for each DC, using the same ***domain-name*** and a new ***domain-controller*** IP.

You can later add one or more child domains to this tree: use the child-domain command described above. The parent-child relationship is established by the domain names: a tree domain of “myco.com” can be the parent of another domain named “mywan.myco.com” or “mylan.myco.com.”

For example, this command sequence adds a tree domain, ‘FDTESTNET.NET’ to the ‘medarcv’ forest.

```
bstnA6k(gbl)# active-directory-forest medarcv
bstnA6k(gbl-forest[medarcv])# tree-domain FDTESTNET.NET 172.16.168.21
bstnA6k(gbl-forest[medarcv])# . . .
```

Removing a Tree Domain

You can remove a tree domain only after all of its child domains have been removed, or if it has at least one redundant DC. The no tree-domain command removes a tree domain from a forest:

no tree-domain *domain-name domain-controller*

For example, this command sequence removes a tree from the ‘medarcv’ forest.

```
bstnA6k(gbl)# active-directory-forest medarcv
bstnA6k(gbl-forest[medarcv])# no tree-domain ALASKA.TEST.NET 172.16.10.137
bstnA6k(gbl-forest[medarcv])# . . .
```

Establishing Forest-to-Forest Trust

You can configure multiple AD forests and, if they have Windows 2003 DCs, establish trust relationships between them. A client from one forest can access a CIFS service in another forest if the forests have a trust relationship. Windows began supporting forest-to-forest trusts in Windows 2003, so this feature is only supported in forests with Windows 2003 DCs.

You must establish the forest-to-forest trust at the DCs before you record it in the ARX configuration.

From gbl mode, use the active-directory forest-trust command to identify a trust relationship between two AD forests:

active-directory forest-trust *forest-a forest-b*

where ***forest-a*** and ***forest-b*** (1-256 characters each) identify the AD forests with the trust relationship. These forests must be pre-configured on the ARX with the active-directory-forest command (recall "Adding an Active-Directory Forest (Kerberos)" on page 3-10), and they must both have all Windows 2003 servers assigned as their forest-roots (see "Identifying the Forest Root" on page 3-10).

This records a two-way trust between the forests. The trust is direct, not transitive; that is, clients in ***forest-a*** can access CIFS services in ***forest-b***, but cannot access any services in ***forest-c*** unless you establish another trust between ***forest-a*** and ***forest-c***. This is consistent with the implementation in Windows.

For example, this command establishes a two-way trust between the 'ny.com' forest and the 'vt.com' forest:

```
bstnA6k(gbl)# active-directory forest-trust ny.com vt.com
bstnA6k(gbl)# . . .
```

Dissolving a Forest-to-Forest Trust

For a forest trust that no-longer exists in your network, you can use the no active-directory forest-trust command:

no active-directory forest-trust *forest-a forest-b*

where ***forest-a*** and ***forest-b*** (1-256 characters each) are the forests where a trust relationship no-longer exists.

For example, this command removes an existing two-way trust between the 'ny.com' and 'ma.com' forests:

```
bstnA6k(gbl)# no active-directory forest-trust ny.com ma.com
bstnA6k(gbl)# . . .
```

Showing All Active-Directory Forests

Use the show active-directory command to show all AD forests and forest trusts on this switch.

show active-directory

For example:

```
bstnA6k(gbl)# show active-directory
Active Directory Domains
------------------------

Forest Name:  medarcv
Domain Name                          Domain Type    IP Address     Service
-----------------------------------  -------------  -------------  ----------
MEDARCH.ORG                          forest-root    192.168.25.102 KDC DNS
MEDARCH.ORG                          forest-root    192.168.25.104 DNS
BOSTONMED.ORG                        tree-domain    172.16.74.88   KDC
FDTESTNET.NET                        tree-domain    172.16.168.21  KDC
BOSTONCIFS.FDTESTNET.NET             child-domain   10.19.230.94   KDC
WESTCOAST.MEDARCH.ORG                child-domain   192.168.202.9  KDC
MA.NE.MEDARCH.ORG                     child-domain   192.168.25.103 KDC
NE.MEDARCH.ORG                       child-domain   172.16.124.73  KDC

Forest Name:  ny.com
Domain Name                          Domain Type    IP Address     Service
-----------------------------------  -------------  -------------  ----------
ny.com                               forest-root    10.52.100.1    KDC
adk.ny.com                           child-domain   10.52.110.1    KDC
catskills.ny.com                     child-domain   10.52.120.1    KDC

Forest Name:  vt.com
Domain Name                          Domain Type    IP Address     Service
-----------------------------------  -------------  -------------  ----------
vt.com                               forest-root    10.52.130.1    KDC
nh.org                               tree-domain    10.52.150.1    KDC
mcniels.vt.com                       child-domain   10.52.140.1    KDC

Forest Trust
------------

Forest-1                             Forest-2                             Trust Type
-----------------------------------  -----------------------------------  ----------
ny.com                               vt.com                               bidirectional
bstnA6k(gbl)# ...
```

Showing One Active-Directory Forest

To focus on a single AD forest, use the forest keyword at the end of the show active-directory command.

show active-directory forest *forest-name*

where ***forest-name*** (1-256 characters) identifies the forest to show.

For example:

```
bstnA6k(gbl)# show active-directory forest medarcv
Active Directory Domains
------------------------


Forest Name:  medarcv
Domain Name                          Domain Type    IP Address     Service
-----------------------------------  -------------  -------------  ----------
MEDARCH.ORG                          forest-root    192.168.25.102 KDC DNS
MEDARCH.ORG                          forest-root    192.168.25.104 DNS
BOSTONMED.ORG                        tree-domain    172.16.74.88   KDC
FDTESTNET.NET                        tree-domain    172.16.168.21  KDC
BOSTONCIFS.FDTESTNET.NET             child-domain   10.19.230.94   KDC
WESTCOAST.MEDARCH.ORG                child-domain   192.168.202.9  KDC
MA.NE.MEDARCH.ORG                     child-domain   192.168.25.103 KDC
NE.MEDARCH.ORG                       child-domain   172.16.124.73  KDC

Forest Trust
------------


Forest-1                             Forest-2                             Trust Type
-----------------------------------  -----------------------------------  ----------
ny.com                               vt.com                               bidirectional
bstnA6k(gbl)# ...
```

Showing One Active-Directory Domain

To focus on a single domain, use the domain keyword at the end of the show active-directory command.

show active-directory domain *domain-name*

where ***domain-name*** (1-256 characters) identifies the domain to show.

For example:

```
bstnA6k(gbl)# show active-directory domain MA.NE.MEDARCH.ORG
Active Directory Domains
------------------------

Forest Name:  medarcv
Domain Name                         Domain Type    IP Address     Service
-----------------------------------  -------------  -------------  ----------
MEDARCH.ORG                         forest-root    192.168.25.104 DNS
MA.NE.MEDARCH.ORG                    child-domain   192.168.25.103 KDC

Forest Trust
------------

Forest-1                            Forest-2                            Trust Type
-----------------------------------  -----------------------------------  ----------
ny.com                              vt.com                              bidirectional
bstnA6k(gbl)# ...
```

Showing DC Status

Use the show active-directory status command to see the status of the AD forest's DCs and all forest-to-forest trusts:

show active-directory status [forest *forest-name* | domain *domain-name*]

where you can use either of the options to focus on a single domain or forest:

forest *forest-name* (optional, 1-256 characters) shows one forest, or

domain *domain-name* (optional, 1-256 characters) focuses on one domain.

For domains with multiple DCs, each ARX processor makes independent decisions about which DC is active. On an ARX®6000, the output shows the status for the SCM processor (1.1) as well as all ASM processors. The SCM processor performs the domain-join operation for an ARX-CIFS service (described in a later chapter about CIFS services), and the ASM processors perform all client authentications. The smaller ARX®500 and ARX®1000 platforms have a single ASM processor that performs both functions. The output of this command shows separate status tables for each processor.

For example:

```
bstnA6k(gbl)# show active-directory status
Processor  1.1:

Transition
Forest        Domain Controller    Domain Name                               Status    Total
Last (UTC)
-------------  -------------------  ----------------------------------------  --------
----  --------------------
vt.com         10.52.140.1          MCNIELS.VT.COM                              Active    1
08:24:38 11/06/2007
vt.com         10.52.150.1          NH.ORG                                      Active    1
08:24:26 11/06/2007
vt.com         10.52.130.1          VT.COM                                      Active    1
08:24:06 11/06/2007
ny.com         10.52.120.1          CATSKILLS.NY.COM                            Active    1
08:23:54 11/06/2007
ny.com         10.52.100.1          NY.COM                                      Active    1
08:23:32 11/06/2007
medarcv        10.19.230.94         BOSTONCIFS.FDTESTNET.NET                    Active    1
08:23:30 11/06/2007
medarcv        172.16.168.21        FDTESTNET.NET                               Active    1
08:23:12 11/06/2007
medarcv        172.16.74.88         BOSTONMED.ORG                               Active    1
08:22:54 11/06/2007
medarcv        192.168.25.103       MA.NE.MEDARCH.ORG                           Active    1
08:22:36 11/06/2007
medarcv        192.168.202.9        WESTCOAST.MEDARCH.ORG                       Active    1
08:22:18 11/06/2007
medarcv        172.16.124.73        NE.MEDARCH.ORG                              Active    1
08:21:59 11/06/2007
medarcv        192.168.25.102       MEDARCH.ORG                                 Active    1
08:21:40 11/06/2007

Forest Trust
------------

Forest-1                  Forest-2                  Trust Type      Last (UTC)
Status
-------------------------  -------------------------  --------------  --------------------
--------------------
ny.com                    vt.com                    bidirectional   08:24:06 11/06/2007
Forest roots are online
```

```
Processor  5.1:

Transition
Forest        Domain Controller    Domain Name                            Status    Total
Last (UTC)
-------------  -------------------  ---------------------------------------  --------
----  --------------------
vt.com        10.52.140.1         MCNIELS.VT.COM                          Active    1
08:24:38 11/06/2007
vt.com        10.52.150.1         NH.ORG                                  Active    1
08:24:26 11/06/2007
vt.com        10.52.130.1         VT.COM                                  Active    1
08:24:06 11/06/2007
ny.com        10.52.120.1         CATSKILLS.NY.COM                        Active    1
08:23:57 11/06/2007
ny.com        10.52.100.1         NY.COM                                  Active    1
08:23:35 11/06/2007
medarcv       10.19.230.94        BOSTONCIFS.FDTESTNET.NET                Active    1
08:23:31 11/06/2007
medarcv       172.16.168.21       FDTESTNET.NET                           Active    1
08:23:12 11/06/2007
medarcv       172.16.74.88        BOSTONMED.ORG                           Active    1
08:22:57 11/06/2007
medarcv       192.168.25.103      MA.NE.MEDARCH.ORG                       Active    1
08:22:39 11/06/2007
medarcv       192.168.202.9       WESTCOAST.MEDARCH.ORG                   Active    1
08:22:21 11/06/2007
medarcv       172.16.124.73       NE.MEDARCH.ORG                          Active    1
08:22:02 11/06/2007
medarcv       192.168.25.102      MEDARCH.ORG                             Active    1
08:21:43 11/06/2007

Forest Trust
------------

Forest-1                  Forest-2                  Trust Type      Last (UTC)
Status
-------------------------  -------------------------  --------------  --------------------
--------------------
ny.com                    vt.com                    bidirectional   08:24:06 11/06/2007
Forest roots are online

Processor  5.2:
```

```
Transition
Forest        Domain Controller    Domain Name                            Status    Total
Last (UTC)
-------------  -------------------  ----------------------------------------  --------
----  --------------------
vt.com         10.52.140.1         MCNIELS.VT.COM                            Active    1
08:24:38 11/06/2007
vt.com         10.52.150.1         NH.ORG                                    Active    1
08:24:26 11/06/2007
vt.com         10.52.130.1         VT.COM                                    Active    1
08:24:09 11/06/2007
ny.com         10.52.120.1         CATSKILLS.NY.COM                          Active    1
08:23:57 11/06/2007
ny.com         10.52.100.1         NY.COM                                    Active    1
08:23:35 11/06/2007
medarcv        10.19.230.94        BOSTONCIFS.FDTESTNET.NET                  Active    1
08:23:31 11/06/2007
medarcv        172.16.168.21       FDTESTNET.NET                             Active    1
08:23:12 11/06/2007
medarcv        172.16.74.88        BOSTONMED.ORG                             Active    1
08:22:57 11/06/2007
medarcv        192.168.25.103      MA.NE.MEDARCH.ORG                         Active    1
08:22:39 11/06/2007
medarcv        192.168.202.9       WESTCOAST.MEDARCH.ORG                     Active    1
08:22:21 11/06/2007
medarcv        172.16.124.73       NE.MEDARCH.ORG                            Active    1
08:22:02 11/06/2007
medarcv        192.168.25.102      MEDARCH.ORG                               Active    1
08:21:43 11/06/2007

Forest Trust
------------

Forest-1                  Forest-2                  Trust Type      Last (UTC)
Status
------------------------- ------------------------- -------------- --------------------
--------------------
ny.com                    vt.com                    bidirectional   08:24:09 11/06/2007
Forest roots are online

bstnA6k(gbl)# ...
```

Focusing On a Single Processor

On an ARX®6000, you can use the optional from clause to focus on a particular processor:

show active-directory status [forest *forest-name* | domain *domain-name*] from *slot.processor*

where

forest *forest-name* and **domain *domain-name*** are described above,

from is a required keyword,

slot (1-6) is the slot number of the desired SCM or ASM, and

processor (1-6) is the processor number. Use show processor for a complete list of processors (and their modules and slots) on the ARX.

For example, this focuses the earlier output on processor 5.2, an ASM processor:

```
bstnA6k(gbl)# show active-directory status from 5.2
Processor  5.2:

Transition
Forest        Domain Controller    Domain Name                           Status    Total
Last (UTC)
-------------  -------------------  ----------------------------------------  --------
----  --------------------
vt.com         10.52.140.1         MCNIELS.VT.COM                          Active    1
08:24:38 11/06/2007
vt.com         10.52.150.1         NH.ORG                                  Active    1
08:24:26 11/06/2007
vt.com         10.52.130.1         VT.COM                                  Active    1
08:24:09 11/06/2007
ny.com         10.52.120.1         CATSKILLS.NY.COM                        Active    1
08:23:57 11/06/2007
ny.com         10.52.110.1         ADK.NY.COM                              Active    1
08:23:35 11/06/2007
ny.com         10.52.100.1         NY.COM                                  Active    1
08:23:35 11/06/2007
medarcv        10.19.230.94        BOSTONCIFS.FDTESTNET.NET                Active    1
08:23:31 11/06/2007
medarcv        172.16.168.21       FDTESTNET.NET                           Active    1
08:23:12 11/06/2007
```

```
medarcv       172.16.74.88       BOSTONMED.ORG                           Active    1
08:22:57 11/06/2007
medarcv       192.168.25.103     MA.NE.MEDARCH.ORG                       Active    1
08:22:39 11/06/2007
medarcv       192.168.202.9      WESTCOAST.MEDARCH.ORG                   Active    1
08:22:21 11/06/2007
medarcv       172.16.124.73      NE.MEDARCH.ORG                          Active    1
08:22:02 11/06/2007
medarcv       192.168.25.102     MEDARCH.ORG                             Active    1
08:21:43 11/06/2007

Forest Trust
------------

Forest-1                  Forest-2                  Trust Type     Last (UTC)
Status
------------------------- ------------------------- -------------- --------------------
--------------------
ny.com                    vt.com                    bidirectional  08:24:09 11/06/2007
Forest roots are online

bstnA6k(gbl)# ...
```

Removing an Active-Directory Forest

You must remove all child domains, tree domains, and the forest-root domain before you can remove an active-directory forest. From gbl mode, use the no active-directory-forest command to delete a forest configuration:

no active-directory-forest *forest-name*

where ***forest-name*** (1-256 characters) identifies the forest to remove.

For example, the following command removes the ‘testkerberos’ forest:

```
bstnA6k(gbl)# no active-directory-forest testkerberos
bstnA6k(gbl)# . . .
```

Authorizing Windows-Management (MMC) Access

You can define a group of Windows clients and their authority to use Windows-management applications, such as the MicroSoft Management Console (MMC). This group can use MMC and similar applications to view or edit CIFS shares, view and/or close open files, or view and/or close open client sessions. You can apply a management-authorization group to any number of CIFS-supporting namespaces (as described in a later chapter).

Use the windows-mgmt-auth command to create a management-authorization group:

windows-mgmt-auth *name*

where ***name*** (1-64 characters) is a name you choose for the group.

This puts you into gbl-mgmt-auth mode, where you enter a list of Windows clients and the management access for them. Once the configuration is finished, you can apply it to any namespace in the clients' Windows domain.

For example, the following command sequence creates a management-authorization group called "readOnly:"

```
bstnA6k(gbl)# windows-mgmt-auth readOnly
bstnA6k(gbl-mgmt-auth[readOnly])# ...
```

Adding a Windows User to the Group

From gbl-mgmt-auth mode, use the user command to add one Windows client to the current management-authorization group:

user *name* windows-domain *domain*

where

name (1-64 characters) is the name of a valid Windows client, and

domain (1-64 characters) is the client's Windows domain.

You can add multiple users to the group; invoke this command once for each user.

For example, the following command sequence adds five users to the management-authorization group, "readOnly:"

```
bstnA6k(gbl)# windows-mgmt-auth readOnly
bstnA6k(gbl-mgmt-auth[readOnly])# user mhoward_md windows-domain MEDARCH.ORG
bstnA6k(gbl-mgmt-auth[readOnly])# user zmarx_md windows-domain MEDARCH.ORG
bstnA6k(gbl-mgmt-auth[readOnly])# user lfine_md windows-domain MEDARCH.ORG
bstnA6k(gbl-mgmt-auth[readOnly])# user choward_md windows-domain MEDARCH.ORG
bstnA6k(gbl-mgmt-auth[readOnly])# user cjderita_md windows-domain MEDARCH.ORG
bstnA6k(gbl-mgmt-auth[readOnly])# ...
```

Removing a User

Use the no user command to remove one user from the current management-authorization group:

no user *name* windows-domain *domain*

where

name (1-64 characters) identifies the user, and

domain (1-64 characters) is the user’s Windows domain.

For example, the following command sequence removes one user from the “readOnly” group:

```
bstnA6k(gbl)# windows-mgmt-auth readOnly
bstnA6k(gbl-mgmt-auth[readOnly])# no user cjderita_md windows-domain MEDARCH.ORG
bstnA6k(gbl-mgmt-auth[readOnly])# ...
```

Setting Management Permissions for the Group

You can set read-only or read-write privileges for any of the following objects in a CIFS namespace:

- CIFS shares,
- CIFS-client sessions, and/or
- open files.

All users in the management-authorization group have the permissions you set with this command. By default, all group members can browse all directories in the namespace, but cannot add or delete CIFS shares. Also, they cannot view or change CIFS-client sessions or open files. From gbl-mgmt-auth mode, use the permit command to enter one permission setting for the current management-authorization group:

permit {share | session | open-file | all} {monitor | any}

where

share | session | open-file | all chooses one type of object (or all of them), and

monitor | any chooses the permissions. The **any** flag allows group members to read and write the object(s); for example, **share any** means that group members can view, add, and delete CIFS shares from the namespace.

Re-use the command to enter more permission settings for this group.

For example, the following command sequence permits the “readOnly” group to view (but not edit) CIFS shares and client sessions, then permits the group to view and/or close open files:

```
bstnA6k(gbl)# windows-mgmt-auth readOnly
bstnA6k(gbl-mgmt-auth[readOnly])# permit share monitor
bstnA6k(gbl-mgmt-auth[readOnly])# permit session monitor
bstnA6k(gbl-mgmt-auth[readOnly])# permit open-file any
bstnA6k(gbl-mgmt-auth[readOnly])# ...
```

Removing a Permission

You can remove access permissions for any or all of the CIFS objects (shares, client sessions, and/or open files). Use the no permit command to remove permissions from the current management-authorization group:

no permit {share | session | open-file | all}

where **share | session | open-file | all** chooses one type of object (or all of them).

For example, the following command sequence removes all open-file permissions from the “readOnly” group:

```
bstnA6k(gbl)# windows-mgmt-auth readOnly
bstnA6k(gbl-mgmt-auth[readOnly])# no permit open-file
bstnA6k(gbl-mgmt-auth[readOnly])# ...
```

Showing All Management-Authorization Groups

You can use the show windows-mgmt-auth command to view all management-authorization groups on the ARX:

show windows-mgmt-auth

For example:

```
bstnA6k(gbl)# show windows-mgmt-auth

Windows Authorization Policy: fullAccess

User Name                                        Domain Name
------------------------------------------------ --------------------------------
juser                                            MEDARCH.ORG
jquser                                           MEDARCH.ORG

Managed Object          Permitted Operation
----------------------- -----------------------
All                     Any


Windows Authorization Policy: readOnly

User Name                                        Domain Name
------------------------------------------------ --------------------------------
mhoward_md                                       MEDARCH.ORG
zmarx_cpa                                        MEDARCH.ORG
lfine_md                                         MEDARCH.ORG
choward_md                                       MEDARCH.ORG

Managed Object          Permitted Operation
```

```
---------------------- -----------------------
Share                  Monitor
Session                Monitor

bstnA6k(gbl)# ...
```

Focusing on One Group

To show a single management-authorization group, add the group name to the end of the show windows-mgmt-auth command:

show windows-mgmt-auth *name*

where ***name*** (1-64 characters) identifies the group to show.

For example:

```
bstnA6k(gbl)# show windows-mgmt-auth readOnly

Windows Authorization Policy: readOnly

User Name                                      Domain Name
---------------------------------------------- -------------------------------
mhoward_md                                     MEDARCH.ORG
zmarx_cpa                                      MEDARCH.ORG
lfine_md                                       MEDARCH.ORG
choward_md                                     MEDARCH.ORG

Managed Object         Permitted Operation
---------------------- -----------------------
Share                  Monitor
Session                Monitor

bstnA6k(gbl)# ...
```

Removing a Management-Authorization Group

You can only remove a management-authorization group if it is not referenced by any namespace. A later chapter describes how to configure a namespace and reference a management-authorization group.

To remove a management-authorization group, use the no windows-mgmt-auth command in gbl mode:

no windows-mgmt-auth *name*

where ***name*** (1-64 characters) identifies the group to remove.

For example, the following command sequence removes a management-authorization group called “beta:”

```
bstnA6k(gbl)# no windows-mgmt-auth beta
bstnA6k(gbl)# ...
```

Chapter 4

Preparing for NFS Authentication

You can create NFS access lists that filter clients based on their IP addresses. You can enter IP addresses directly and/or refer to pre-defined netgroups at a Network Information Service (NIS) server. A NIS *netgroup* defines a group of host machines, and may also contain other NIS netgroups.

This chapter pertains to NFS-client authentication only; you can skip this chapter unless you plan to offer NFS service with some level of client authentication.

Before You Begin

A NIS netgroup often refers to hosts by their DNS hostnames rather than their IP addresses. The ARX must use the local DNS server(s) to translate those names into IP addresses. Before you start configuring NIS domains, configure the switch to perform DNS lookups. Refer to the *CLI Network-Management Guide*: see "Configuring DNS Lookups" on page 4-35.

All NIS servers used by the switch must also use the same DNS server(s). If this switch has a redundant peer, it must also use the same DNS server(s).

Adding a NIS Domain

The first step in using NIS netgroups is configuring a NIS domain on the switch. You can skip to the next section ("Adding an NFS Access List" on page 4-9) if you do not plan to use any NIS netgroups in your NFS access lists.

The switch supports up to eight NIS domains. From gbl mode, use the nis domain command to add a new one:

nis domain *domain*

where ***domain*** (1-256 characters) is the name of the domain (for example, “acopia” in “server.acopia.com”). This must match the name of the NIS domain defined in one or more external NIS servers.

This places you into gbl-nis-dom mode, where you identify one or more NIS servers that host this NIS domain.

For example, the following command sequence adds a NIS domain named “wwmed.com:”

```
bstnA6k(gbl)# nis domain wwmed.com
bstnA6k(gbl-nis-dom[wwmed.com])# ...
```

Identifying a NIS Server for the NIS Domain

The final step in configuring an NIS domain is to identify one or more NIS servers that host the domain. From gbl-nis-dom mode, use the ip address command to identify one server:

ip address *ip-address*

where ***ip-address*** is in dotted-decimal format (for example, 192.168.25.122).

You can configure up to four NIS servers per NIS domain. They are used in the order that they are defined. If the first server times out or is unreachable, the ARX tries to connect to the next one, and so on.

For example, the following command identifies three NIS servers at 192.168.25.203, 204, and 205:

```
bstnA6k(gbl)# nis domain wwmed.com
bstnA6k(gbl-nis-dom[wwmed.com])# ip address 192.168.25.203
bstnA6k(gbl-nis-dom[wwmed.com])# ip address 192.168.25.204
bstnA6k(gbl-nis-dom[wwmed.com])# ip address 192.168.25.205
bstnA6k(gbl-nis-dom[wwmed.com])# ...
```

Removing a NIS Server

Use the no ip address command to remove one of the NIS servers from the list:

no ip address *ip-address*

where ***ip-address*** is in dotted-decimal format (for example, 192.168.25.122).

Caution

If you remove the only NIS server for the current NIS domain, support for the domain is limited. The switch keeps a local cache with all NIS netgroups, but can never refresh this cache. On the next switch reboot, the switch clears its NIS cache: this removes NIS support altogether. This can cause serious access issues for NFS clients that belong to the domain's netgroups.

For example, the following command removes one NIS server from the wwmed.com domain:

```
bstnA6k(gbl)# nis domain wwmed.com
bstnA6k(gbl-nis-dom[wwmed.com])# no ip address 192.168.25.205
bstnA6k(gbl-nis-dom[wwmed.com])# ...
```

Listing All Configured NIS Domains

You can use the show nis domain command to get a list of all configured NIS domains on the ARX:

show nis domain

For example, this command shows a single NIS domain backed by three servers:

```
bstnA6k(gbl)# show nis domain

NIS Domain                        Last Update   Status    Servers
--------------------------------  ------------  --------  ---------------
wwmed.com                         26 Jan 03:24  Success   192.168.25.201
                                                          192.168.25.204
                                                          192.168.25.205

bstnA6k(gbl)# ...
```

Showing Details for a NIS Domain

Add the name of an NIS domain to show details:

show nis domain *name*

where ***name*** (1-256 characters) identifies the NIS domain.

For example:

```
bstnA6k(gbl)# show nis domain wwmed.com

NIS Domain:                      wwmed.com
Server(s):                       192.168.25.201
                                 192.168.25.204
                                 192.168.25.205
Last Update:                     Fri Jan 26 03:24:16 2007
Last Update Status:              Success
Last Successful Update:          Fri Jan 26 03:24:16 2007
Netgroups:                       2397
Netgroup Resolution Errors:      0
Hosts:                           48046
Hosts Resolved:                  47507

bstnA6k(gbl)# ...
```

Listing Netgroups in a NIS Domain

You can use the show nis netgroup command to query the NIS-domain server for a list of the netgroups in the domain:

show nis netgroup *domain*

where ***domain*** (1-256 characters) identifies the NIS domain.

For example:

```
bstnA6k(gbl)# show nis netgroup wwmed.com

Netgroup
---------------------------------------------------------------------
```

```
auto_1
...
medtechs
surgeons

Total Netgroups: 2396
bstnA6k(gbl)# ...
```

Showing the Members of One Netgroup

For a list of members in a NIS netgroup, add the name of the netgroup to the end of the show nis netgroup command:

show nis netgroup *domain netgroup*

where

domain (1-256 characters) identifies the NIS domain, and

netgroup (1-1024 characters) is the specific netgroup.

This shows the host machines in the netgroup, along with their IP addresses. The ARX does not authenticate users and groups, so it does not use those IDs from the netgroup. The back-end filers authenticate users and groups, and the switch either allows or disallows client access based on the result.

For example, this shows the host machines in the “medtechs” netgroup:

```
bstnA6k(gbl)# show nis netgroup wwmed.com medtechs

Netgroup successfully resolved.

Hostname                                            IP Address
--------------------------------------------------- -----------------
bench2.wwmed.com                                    10.51.201.72
bench3.wwmed.com                                    10.51.201.73
bench4.wwmed.com                                    10.51.201.74

Total Resolved Hosts: 3
bstnA6k(gbl)# ...
```

Updating the NIS Database

The ARX keeps an internal copy of all the NIS netgroups and their fully-resolved hosts. The database is built when you add the NIS domain to the switch; it is used for switch operation as well as the show commands above. To avoid excessive traffic to the DNS server, the switch does not update this database automatically. You can rebuild the database manually after any large-scale DNS or NIS changes.

From priv-exec mode, use the nis update command to update the netgroup database:

nis update [*nis-domain*]

where ***nis-domain*** (optional: 1-256 characters) causes the update to focus on a single NIS domain.

In a redundant pair of switches, this triggers concurrent NIS updates from both peers. Since they keep independent NIS caches, failovers do not incur any additional downtime for NIS.

The show nis domain command (above) shows the time of the most-recent update.

For example, the following command sequence updates all NIS domains:

```
bstnA6k(gbl)# end

bstnA6k# nis update

bstnA6k# ...
```

Reading the Update Report

Each NIS update creates one or more reports (one per updated domain) to show its results. The NIS software names the reports according to the following convention:

```
nis-update.domain-name.rpt
```

where *domain-name* is the name of the NIS domain in the report.

To conserve disk space, each NIS update overwrites any previous reports.

Use the show reports command to list all reports, including the NIS-update reports.

show reports

Use show reports *report-name*, tail, or grep to read the file. To save the report off to an external FTP site, use the copy ... ftp command from priv-exec mode. To upload the report to an SCP host, use copy ... scp. All of these commands are documented in the *CLI Reference* manual.

This report lists all the hosts in the NIS domain that had issues, such as the name not being found at the DNS server. You can use this report as a guide to adjust the DNS and/or NIS configurations on the back-end servers.

For example, this shows a NIS-update report for a large domain, “wwmed.com:”

```
bstnA6k(gbl)# show reports nis-update.wwmed.com.rpt
**** NIS Update Report: Started at Wed Dec  7 09:45:03 2005 ****
**** NIS Domain: wwmed.com
**** NIS Server: 192.168.25.203
**** NIS Server: 192.168.25.204
**** NIS Server: 192.168.25.205

**** Legend:
****   HN = Hostname not found.
****   NP = Netgroup parsing error.
****   NG = Netgroup not found.
****   WG = Watched netgroup has changed contents.
****   NE = NIS server error.

Status            Hostname/Netgroup
----------------  -------------------------------------------------------------
[HN             ]  not_a_real_host1 in group: bad_hosts
[HN             ]  not_a_real_host2 in group: bad_hosts
[HN             ]  not_a_real_host3 in group: bad_hosts
[HN             ]  not_a_real_host4 in group: bad_hosts
[HN             ]  not_a_real_host5 in group: bad_hosts
[HN             ]  baghdad in group: chassis
[HN             ]  ommegang in group: chassis
[HN             ]  bismark in group: chassis

...
```

```
[HN            ]  london in group: sixthousands
[HN            ]  montreal in group: sixthousands
[HN            ]  lasvegas in group: sixthousands

Netgroups Processed:                           2,396
Hosts Processed:                              48,043
Hostnames Not Found:                             539
Netgroup Parsing Errors:                           0
Netgroups Not Found:                               0
Watched Netgroup Changes                           0

**** Elapsed time:           00:00:17
**** NIS Update Report: DONE at Wed Dec  7 09:45:20 2005 ****
bstnA6k(gbl)# ...
```

Scheduling Regular Updates

If your site makes smaller, incremental changes to DNS and/or NIS netgroups, you can use the at command to arrange for regular NIS updates. The at command is in cfg mode.

As mentioned above, a redundant pair keeps independent NIS caches on each peer. Schedule these updates on both peers.

For example, the following command sequence runs the nis update command every night at 5:15 AM:

```
bstnA6k(gbl)# end
bstnA6k# config
bstnA6k(cfg)# at 05:15:00 every 1 days do “nis update”
bstnA6k(cfg)# ...
```

As another example, the following command sequence causes nis update to run every Saturday at midnight:

```
bstnA6k(gbl)# end
bstnA6k# config
bstnA6k(cfg)# at 00:00:00 every saturday do “nis update”
bstnA6k(cfg)# ...
```

Removing the NIS Domain-Server Map

From gbl mode, use no nis domain to remove a NIS domain-server map:

no nis domain *domain*

where ***domain*** (1-256 characters) is the name of the domain to remove.

You cannot remove a domain that is referenced by an NFS access list. The next section describes how to use an NIS domain in an access list.

For example, the following command sequence removes the NIS domain named “PROVIDENCE:”

```
bstnA6k(gbl)# no nis domain PROVIDENCE
bstnA6k(gbl)# ...
```

Adding an NFS Access List

Before you configure any NFS exports or services, you can optionally configure one or more NFS access lists. An *access list* is a list of subnets and hosts to which you permit or deny access to NFS service. For example, you could permit access from the subnet at 192.168.101.0 but deny access from all other subnets.

Note

If you currently use NIS at your back-end filers, the front-end NFS service passes the client’s User ID and Group ID through to the back-end filer, and the back-end filer authenticates against those IDs as usual. The access list filters out users based on IP address only, but the filer(s) may deny access based on user ID after the IP address passes at the ARX.

If you are not planning to configure any NFS shares or services, you can skip this section. Otherwise, when you configure NFS shares and services later, you can apply one NFS access list to each NFS share. You can reuse one NFS access list for any number of NFS shares.

The ARX supports up to 512 NFS access lists. From gbl mode, use the nfs-access-list command to create a new one:

nfs-access-list *list-name*

where ***list-name*** (1-64 characters) is a name you choose for the access list.

The CLI prompts for confirmation before creating the new NFS access list. Enter **yes** to proceed. This places you in gbl-nfs-acl mode, from which you can configure various permit and deny rules for specific subnets and/or NIS netgroups. By default, all subnets and netgroups are denied any access.

For example, the following command sequence creates a new NFS access list:

```
bstnA6k(gbl)# nfs-access-list eastcoast


This will create a new NFS ACL.


Create NFS ACL ''eastcoast''? [yes/no] yes
bstnA6k(gbl-nfs-acl[eastcoast])# ...
```

Listing All NFS Access Lists

To verify that your NFS access list was added to the configuration, use the show nfs-access-list command. You can invoke this command from any mode.

show nfs-access-list

The output shows all configured access lists in a table, one access list per row. For each access list, high-level details are shown. These details are clarified in the sections below.

For example:

```
bstnA6k(gbl)# show nfs-access-list
Access List Name    Anon UID Anon GID    Num Rules      Num References
  ------------------------------------------------------------------------
  eastcoast           100      100                8                   2
  westcoast           65534    65534              2                   0
bstnA6k(gbl)# ...
```

Showing One NFS Access List

As you configure your NFS access lists, it will be convenient to see the current list settings. Use the show nfs-access-list command with the specific access-list name to see the full configuration for one access list:

show nfs-access-list *list-name*

where ***list-name*** (1-64 characters) identifies the access list.

Among other configuration details, the output shows the order of all permit and deny rules in the access list. As explained below, the order of these rules is important.

For example:

```
bstnA6k(gbl)# show nfs-access-list eastcoast

Access List Name: eastcoast
  Description:                allowable subnets in MA, NY, & DC
  NIS Domain:                 wwmed.com
  Anonymous UID:              100
  Anonymous GID:              100
  Number of References:       2

permit 172.16.100.0 255.255.255.0 read-write root squash
permit 172.16.204.0 255.255.255.0 read-only  root allow
permit 172.16.0.0 255.255.0.0 read-write root squash
permit netgroup surgeons read-write root allow
permit netgroup medtechs read-only  root squash
deny  192.168.77.0 255.255.255.0
deny  192.168.202.0 255.255.255.0
permit 192.168.0.0 255.255.0.0 read-write root squash

bstnA6k(gbl)# ...
```

Resolving All Netgroups in the Access List

If the access list contains any netgroups, you can resolve those netgroups to see all of the hosts within them. To accomplish this, add the resolve-netgroups keyword to the end of the command:

show nfs-access-list *list-name* resolve-netgroups

This provides a complete view of the access list, resolving all netgroups to the IP addresses for their hosts. For example, this expands the previous output to show all hosts in the “surgeons” and “medtechs” netgroups:

```
bstnA6k(gbl)# show nfs-access-list eastcoast resolve-netgroups


Access List Name: eastcoast
  Description:              allowable subnets in MA, NY, & DC
  NIS Domain:               wwmed.com
  Anonymous UID:            100
  Anonymous GID:            100
  Number of References:     2

permit 172.16.100.0 255.255.255.0 read-write root squash
permit 172.16.204.0 255.255.255.0 read-only  root allow
permit 172.16.0.0 255.255.0.0 read-write root squash
permit 10.51.201.71 255.255.255.255 read-write root allow
permit 10.51.201.72 255.255.255.255 read-only  root squash
permit 10.51.201.73 255.255.255.255 read-only  root squash
permit 10.51.201.74 255.255.255.255 read-only  root squash
deny  192.168.77.0 255.255.255.0
deny  192.168.202.0 255.255.255.0
permit 192.168.0.0 255.255.0.0 read-write root squash

Number of entries in access list: 10
All Netgroup(s) were successfully resolved.

bstnA6k(gbl)# ...
```

Each access list can support a maximum of 2048 permit and deny rules, including the individual permit rules for every host in every netgroup. If you exceed the limit (perhaps because of an overly-large netgroup), this output shows the first 2048 entries followed by an error.

Setting the NIS Domain

If the access list will use NIS netgroups, you must set the access list's NIS domain. The ARX needs the NIS domain to access the local NIS server. (You map the NIS domain name to an NIS server ahead of time, as described earlier: recall "Adding a NIS Domain" on page 4-1.)

From gbl-nfs-acl mode, use the nis domain command to set the NIS domain for the access list:

nis domain *domain*

where ***domain*** can be up to 256 characters long.

For example, the following command sequence lists all NIS domains, shows only "wwmed.com," and uses it in an access list named "eastcoast:"

```
bstnA6k(gbl)# show nis domain

NIS Domain                        Last Update   Status    Servers
--------------------------------  ------------  --------  ---------------
wwmed.com                         26 Jan 03:24  Success   192.168.25.201
                                                          192.168.25.204
                                                          192.168.25.205

bstnA6k(gbl)# nfs-access-list eastcoast
bstnA6k(gbl-nfs-acl[eastcoast])# nis domain wwmed.com
bstnA6k(gbl-nfs-acl[eastcoast])# ...
```

Removing the NIS Domain

Use no nis domain to remove the NIS domain from the access list:

no nis domain

For example:

```
bstnA6k(gbl)# nfs-access-list westcoast
bstnA6k(gbl-nfs-acl[eastcoast])# no nis domain snemed.com
bstnA6k(gbl-nfs-acl[eastcoast])# ...
```

Adding a Permit Rule

By default, a new NFS access list denies access to all subnets. You can selectively allow access by configuring a *permit* rule for each trusted subnet. From gbl-nfs-acl mode, use the permit command to add a permit rule for one subnet:

permit *ip-address mask* [read-only]

where

ip-address is the address of the subnet,

mask defines the network part of the *ip-address*, and

read-only is an optional flag to permit read access but deny writes. If omitted, this defaults to allowing both reads and writes.

For example, the following command sequence permits read-write access to clients at 172.16.100.0:

```
bstnA6k(gbl)# nfs-access-list eastcoast
bstnA6k(gbl-nfs-acl[eastcoast])# permit 172.16.100.0 255.255.255.0
bstnA6k(gbl-nfs-acl[eastcoast])# ...
```

Permitting a Netgroup

If you have configured a NIS domain for this access list (see above), you can refer to a netgroup configured in that domain. This leverages any netgroups that were configured before the introduction of the ARX. From gbl-nfs-acl mode, use the permit netgroup command to permit a NIS netgroup:

permit netgroup *name* [read-only]

where

name (1-1024 characters) identifies a netgroup from the NIS domain, and

read-only is an optional flag to permit read access but deny writes. If omitted, this defaults to allowing both reads and writes.

This permits all hosts in the netgroup. As explained earlier, the netgroups users and/or groups are handled at the back-end filer(s), not the switch.

For example, the following command sequence shows all netgroups in the "wwmed.com" domain and permits read-only access to host machines in the "medtechs" netgroup:

```
bstnA6k(gbl)# show nis netgroup wwmed.com

Netgroup
--------------------------------------------------------------------
auto_1
...
medtechs
surgeons

Total Netgroups: 2396
bstnA6k(gbl)# nfs-access-list eastcoast
bstnA6k(gbl-nfs-acl[eastcoast])# permit netgroup medtechs read-only
bstnA6k(gbl-nfs-acl[eastcoast])# ...
```

Rule Ordering

The order of rules is very important in an access list. Whenever a client tries to access an NFS service with an access list, the client's IP address is compared to the rules in the order they were entered. If the IP address matches two rules, the first rule is used and the second rule is ignored.

For example, consider the two permit rules below. Clients in 192.168.10.x would match the first rule, while clients outside that subnet in the same Class-B network (192.168.x.x) would match the second rule. This would give read-only access to clients in the Class-B network but full read-write access clients in the smaller Class-C subnet:

```
permit 192.168.10.0 255.255.255.0

permit 192.168.0.0 255.255.0.0 read-only
```

If the rules were reversed, clients in the Class-C subnet would match the read-only rule before reaching the read-write rule that was intended for them.

Allowing Root Access

A new permit rule squashes root access by default. That is, if a client logs in as the *root* user (sometimes called the superuser) and accesses the NFS share, the ARX translates the client's user ID to an *anonymous* ID with very low access privileges. The client can therefore write only to files or directories with wide-open permission settings. This is the safest strategy for a permit rule, as it prevents *root* users from damaging NFS shares.

You have the option to disable root squashing in a permit rule. From gbl-nfs-acl mode, use the root allow keywords at the end of the permit command to allow root access:

> **permit *ip-address mask* [read-only] root allow**
>
> or
>
> **permit netgroup *name* [read-only] root allow**

Caution

This setting permits clients with *root* access to change or remove any (or all) files or directories. Whether by accident or malicious intent, this could result in loss or corruption in client data.

For example, the following command sequence allows *root* access from clients at 172.16.204.0. To control the security problem, access is read-only for this rule:

```
bstnA6k(gbl)# nfs-access-list eastcoast
bstnA6k(gbl-nfs-acl[eastcoast])# permit 172.16.204.0 255.255.255.0 read-only root allow
bstnA6k(gbl-nfs-acl[eastcoast])# ...
```

As another example, this command sequence allows *root* access from clients in the “surgeons” netgroup:

```
bstnA6k(gbl)# nfs-access-list eastcoast
bstnA6k(gbl-nfs-acl[eastcoast])# permit netgroup surgeons root allow
bstnA6k(gbl-nfs-acl[eastcoast])# ...
```

Removing a Permit Rule

From gbl-nfs-acl mode, use no permit to remove a permit rule from the current access list:

no permit *ip-address mask*

where:

ip-address identifies the subnet for the permit rule, and

mask defines the network part of the *ip-address*.

or

no permit netgroup *name*

where ***name*** (1-1024 characters) identifies the netgroup to remove.

For example, the following command sequence removes the permit rule for clients at 172.16.13.0:

```
bstnA6k(gbl)# nfs-access-list eastcoast
bstnA6k(gbl-nfs-acl[eastcoast])# no permit 172.16.13.0 255.255.255.0
bstnA6k(gbl-nfs-acl[eastcoast])# ...
```

Changing the Anonymous User ID

When permit rules have root-squash enabled, they translate the User ID (UID) of a *root* user to an *anonymous* UID. By default, the access list uses 65534 for this UID. To change the UID for *anonymous*, use the anonymous-uid command:

anonymous-uid *id*

where ***id*** is a number from 1-65535.

For example, the following command sequence sets the *anonymous* UID to 100:

```
bstnA6k(gbl)# nfs-access-list eastcoast
bstnA6k(gbl-nfs-acl[eastcoast])# anonymous-uid 100
bstnA6k(gbl-nfs-acl[eastcoast])# ...
```

Changing the Anonymous Group ID

When permit rules have root-squash enabled, they translate the Group ID (GID) of a *root* user to an *anonymous* GID. By default, the access list uses 65534. To change the GID for *anonymous*, use the anonymous-gid command:

anonymous-gid *id*

where ***id*** is a number from 1-65535.

For example, the following command sequence sets the *anonymous* GID to 100:

```
bstnA6k(gbl)# nfs-access-list eastcoast
bstnA6k(gbl-nfs-acl[eastcoast])# anonymous-gid 100
bstnA6k(gbl-nfs-acl[eastcoast])# ...
```

Reverting to the Default Group ID

For root squashing, an access list uses the default GID of 65534. To revert to this default, use the no anonymous-gid command:

no anonymous-gid

For example:

```
bstnA6k(gbl)# nfs-access-list westcoast
bstnA6k(gbl-nfs-acl[westcoast])# no anonymous-gid
```

```
bstnA6k(gbl-nfs-acl[westcoast])# ...
```

Reverting to the Default User ID

As with the GID, an access list uses the default UID of 65534 when it performs root squashing. From gbl-nfs-acl mode, use the no anonymous-uid command to revert to this default:

no anonymous-uid

For example:

```
bstnA6k(gbl)# nfs-access-list westcoast
bstnA6k(gbl-nfs-acl[westcoast])# no anonymous-uid
bstnA6k(gbl-nfs-acl[westcoast])# ...
```

Adding a Deny Rule

You may have a situation where most of a large subnet should be permitted access to NFS, but some portions of the subnet should be denied access. From gbl-nfs-acl mode, use the deny command to add a deny rule for one subnet:

deny *ip-address mask*

where

ip-address is the address of the subnet, and

mask defines the network part of the *ip-address*.

For example, the following command sequence denies access to two Class-C subnets, but then permits access to any IP outside those subnets but *inside* their Class-B supernet, 192.168.0.0/16:

```
bstnA6k(gbl)# nfs-access-list eastcoast
bstnA6k(gbl-nfs-acl[eastcoast])# deny 192.168.77.0 255.255.255.0
bstnA6k(gbl-nfs-acl[eastcoast])# deny 192.168.202.0 255.255.255.0
bstnA6k(gbl-nfs-acl[eastcoast])# permit 192.168.0.0 255.255.0.0
bstnA6k(gbl-nfs-acl[eastcoast])# ...
```

You cannot deny a NIS netgroup. We recommend a subnet-deny rule after any permit netgroup rule, to ensure that all other hosts in the netgroup's subnet are explicitly denied.

Removing a Deny Rule

From gbl-nfs-acl mode, use no deny to remove a deny rule from the current access list:

no deny *ip-address mask*

where

ip-address identifies the subnet for the deny rule, and

mask defines the network part of the *ip-address*.

For example, the following command sequence removes the deny rule for clients at 192.168.77.0:

```
bstnA6k(gbl)# nfs-access-list eastcoast
bstnA6k(gbl-nfs-acl[eastcoast])# no deny 192.168.77.0 255.255.255.0
bstnA6k(gbl-nfs-acl[eastcoast])# ...
```

Changing Rule Order

The order that you enter rules determines the order in which they are compared to client IPs. If a client's IP address matches more than one rule in the access list, the ARX uses the first matching rule and ignores the rest. For example, consider the following access list:

```
bstnA6k(gbl)# show nfs-access-list eastcoast


...
permit 192.168.0.0 255.255.0.0 read-write root squash
deny  192.168.77.0 255.255.255.0
deny  192.168.202.0 255.255.255.0
...
```

These permit and deny rules have a subtle configuration error. The intention was to allow all clients from 192.168.0.0 *except* clients from 192.168.77.0 or 192.168.202.0. For example, a client at IP 192.168.77.29 is supposed to be blocked by the first deny rule, “deny 192.168.77.0 ...” However, that IP address matches the Class-B network (192.168.0.0) in the earlier permit rule. The deny rules can never actually be reached.

To correct this configuration error, you must delete the rule(s) that is/are out of order, then add rules back into the access list in the correct order. This re-ordering method conforms to industry standards for configuring access lists.

The following command sequence shows an NFS access list with this mis-configuration and then corrects it:

```
bstnA6k(gbl)# show nfs-access-list eastcoast

Access List Name: eastcoast
  Description:               allowable subnets in MA, NY, & DC
  NIS Domain:                wwmed.com
  Anonymous UID:             100
  Anonymous GID:             100
  Number of References:      1

permit 172.16.100.0 255.255.255.0 read-write root squash
permit 172.16.204.0 255.255.255.0 read-only  root allow
permit 172.16.0.0 255.255.0.0 read-write root squash
permit netgroup surgeons read-write root allow
permit netgroup medtechs read-only  root squash
permit 192.168.0.0 255.255.0.0 read-write root squash
deny  192.168.77.0 255.255.255.0
deny  192.168.202.0 255.255.255.0
```

First, remove the permit rule and show that it is gone:

```
bstnA6k(gbl)# nfs-access-list eastcoast
bstnA6k(gbl-nfs-acl[eastcoast])# no permit 192.168.0.0 255.255.0.0
bstnA6k(gbl-nfs-acl[eastcoast])# show nfs-access-list eastcoast
...
permit netgroup surgeons read-write root allow
permit netgroup medtechs read-only  root squash
deny  192.168.77.0 255.255.255.0
deny  192.168.202.0 255.255.255.0
```

Add back the permit rule and show that it is now at the end of the list:

```
bstnA6k(gbl-nfs-acl[eastcoast])# permit 192.168.0.0 255.255.0.0
bstnA6k(gbl-nfs-acl[eastcoast])# show nfs-access-list eastcoast
...
deny  192.168.77.0 255.255.255.0
deny  192.168.202.0 255.255.255.0
permit 192.168.0.0 255.255.0.0 read-write root squash
bstnA6k(gbl-nfs-acl[eastcoast])# ...
```

Adding a Description

You can add a description to the access list for use in the show command. The description can differentiate the list from others. From gbl-nfs-acl mode, use the description command to add a description:

description *text*

where ***text*** is 1-255 characters. Quote the text if it contains any spaces.

For example:

```
bstnA6k(gbl)# nfs-access-list eastcoast
bstnA6k(gbl-nfs-acl[eastcoast])# description "allowable subnets in MA, NY, & DC"
bstnA6k(gbl-nfs-acl[eastcoast])# ...
```

Removing the Description

From gbl-nfs-acl mode, use no description to remove the description string from the current access list:

no description

For example:

```
bstnA6k(gbl)# nfs-access-list westcoast
bstnA6k(gbl-nfs-acl[westcoast])# no description
bstnA6k(gbl-nfs-acl[westcoast])# ...
```

Removing an Access List

From gbl mode, use the no nfs-access-list command to remove an NFS access list:

no nfs-access-list *list-name*

where ***list-name*** (1-64 characters) identifies the access list to remove.

You must remove all references to the access list before you can use this command to remove the list itself. An access list is referenced from an NFS service; instructions for configuring an NFS service appear below.

For example, the following command sequence removes the NFS access list, "southwest:"

```
bstnA6k(gbl)# no nfs-access-list southwest
bstnA6k(gbl)# ...
```

Chapter 5

Examining Filers

Use the show exports and probe exports commands to examine filers in the server network. These commands make queries from proxy-IP addresses to test filer connectivity, find the services supported by the filer, discover filer shares, and discover permissions settings at the shares. You can use them to troubleshoot network connectivity to filers as well as any permissions issues.

From any mode, use show exports to examine a filer's shares and capabilities:

show exports host *filer*

where ***filer*** (1-1024 characters) is the filer's IP address or host name. You can use the hostname only if the switch is configured for DNS lookups; refer to "Configuring DNS Lookups" on page 4-35 in the *CLI Network-Management Guide*.

This shows a report with four tables:

- Connectivity - shows whether or not pings succeed from every active proxy-IP address. All proxy-IP pings must succeed. If they fail, verify that the filer is on the proxy-IP subnet ("Adding a Range of Proxy-IP Addresses" on page 4-6 of the *CLI Network-Management Guide*) or reachable through a gateway on that subnet (via static route: see "Adding a Static Route" on page 4-9 of the network guide). If some succeed and some fail for a given proxy IP, the MTU settings are probably set inconsistently between the proxy IP, the filer, and some of the network equipment between them. For a command to change the MTU on an ARX VLAN, refer to "Enabling Jumbo Frames (optional)" on page 3-6 of the network guide.

- Capabilities - shows the transport protocols (TCP or UDP) and port numbers for NFS and CIFS. For NFS, this shows the same information for portmapper and the mount daemon: an NFS filer must support all three services. For CIFS servers, this also shows the server-level security settings.
- Shares - For NFS, this shows the Path of the export, the results of a UNIX **showmount -e** command (which may be blank, depending on the text string that the filer returns), and the Status of a test mount from the ARX. The test mount must succeed for the share to be usable in a namespace.

  For CIFS, this shows each Share name, the Total and Free storage space on the share, and the Serial Num of the storage volume behind the share. Note that the space measures apply to the storage volume behind the share; if multiple shares map to the same serial number (and therefore the same storage volume), their space measures are all the same.
- Time - this shows the time skew between the ARX and the filer, if any. Namespace policy and Kerberos authentication (described in later chapters) require that the clocks be nearly synchronized between the ARX and its filers; NTP is recommended. If the filer has both CIFS and NFS shares, there may be different skews for each server.

For example, the following command shows all four tables for a filer at 192.168.25.19:

```
bstnA6k> show exports host 192.168.25.19
Export probe of filer "192.168.25.19"

Connectivity:

  Slot.Proc  Proxy Address    Ping (size: result)
  ---------  ---------------  -------------------
  3.1        192.168.25.31    64: Success  2000: Success  8820: Success
  3.2        192.168.25.32    64: Success  2000: Success  8820: Success
  3.3        192.168.25.33    64: Success  2000: Success  8820: Success
  3.4        192.168.25.34    64: Success  2000: Success  8820: Success
  3.5        192.168.25.55    64: Success  2000: Success  8820: Success
  3.6        192.168.25.56    64: Success  2000: Success  8820: Success

CIFS Credentials: [anonymous]
```

```
Capabilities:
  NFS
    Port Mapper    TCP/111, UDP/111
    Mount Daemon   V1 TCP/1016, V1 UDP/1013, V2 TCP/1016, V2 UDP/1013, V3
TCP/1016, V3 UDP/1013
    Server         V2 TCP/2049, V2 UDP/2049, V3 TCP/2049, V3 UDP/2049

  CIFS
    Security Mode  User level, Challenge/response, Signatures disabled
    Server         TCP/445
    Max Request    16644 bytes

Shares:
  NFS
    Path (Owner)                        Access (Status)
    ----------------------------------  ---------------------------
    /disk2                              * (Mounted,rsize=32768,wsize=32768)
    /exports                            * (Mounted,rsize=32768,wsize=32768)

  CIFS
                                         Storage Space
    Share                              Total (MB)   Free (MB)    Serial Num
    -------------------------------   ----------  ----------    ----------
    cifs
    INFO: CIFS reported STATUS_ACCESS_DENIED on share cifs.

Time:
  NFS
    Filer's time is the same as the switch's time.

  CIFS
    Filer's time is the same as the switch's time.
bstnA6k> ...
```

Examining CIFS Shares

You can only examine CIFS shares if you have sufficient permissions at the filer. Use the user and windows-domain options to provide Windows credentials to the filer:

show exports host *filer* user *username* windows-domain *domain*

where

username (1-64 characters) is the username,

domain (1-64 characters) is the user's Windows domain, and

the other options are explained above.

The CLI prompts for the user's password. Enter the password to continue. The command only shows CIFS shares, capabilities, and time.

Note

The filer queries can take more than one minute for a filer with a large number of CIFS shares.

The CLI only shows CIFS shares and other Windows-related information when you provide the Windows username.

For example, this is a CIFS-side probe of a filer at 192.168.25.21.

```
bstnA6k> show exports host 192.168.25.21 user jqpublic windows-domain MEDARCH
Password: jqpasswd
Export probe of filer "192.168.25.21"

Connectivity:

  Slot.Proc  Proxy Address    Ping (size: result)
  ---------  ---------------  --------------------
  3.1        192.168.25.31    64: Success  2000: Success  8820: Success
  3.2        192.168.25.32    64: Success  2000: Success  8820: Success
  3.3        192.168.25.33    64: Success  2000: Success  8820: Success
  3.4        192.168.25.34    64: Success  2000: Success  8820: Success
  3.5        192.168.25.55    64: Success  2000: Success  8820: Success
  3.6        192.168.25.56    64: Success  2000: Success  8820: Success

CIFS Credentials: MEDARCH\jqpublic

Capabilities:

  CIFS
    Security Mode  User level, Challenge/response, Signatures disabled
    Server         TCP/139
    Max Request    16644 bytes

Shares:

  CIFS
                                        Storage Space
    Share                             Total (MB)   Free (MB)    Serial Num
    -------------------------------   ----------  ----------    ----------
...
    insurance                             174367       30173     f144-cf04
...
    -------------------------------   ----------  ----------    ----------
    Share                             Total (MB)   Free (MB)    Serial Num
                                        Storage Space

Time:

  CIFS
    Filer's time is the same as the switch's time.
bstnA6k> ...
```

Using Proxy-User Credentials

If there is a proxy user that is already configured for the filer's domain, you can use the proxy user configuration instead of a username, domain, and password. Use show proxy-user for a full list of all proxy users (recall "Listing All Proxy Users" on page 3-5). The CLI does not prompt for a password if you use a pre-configured proxy user:

show exports host *filer* proxy-user *proxy*

where

proxy (1-64 characters) is the name of the proxy-user configuration, and

the other options are explained above.

As with the user/windows-domain syntax, this limits the output of the command to CIFS shares, capabilities, and time readings.

For example, this command sequence uses a set of proxy-user credentials to examine the filer at 192.168.25.20:

```
bstnA6k> show proxy-user

Name                                Domain            User
-------------------------------------------------------------------------------
acoProxy1                           WWMEDNET          jqprivate
acoProxy3                           FDTESTNET         jqtester
acoProxy2                           BOSTONCIFS        jqpublic
bstnA6k> show exports host 192.168.25.20 proxy-user acoProxy2
Export probe of filer "192.168.25.20"

Connectivity:

  Slot.Proc  Proxy Address    Ping (size: result)
  ---------  ---------------  --------------------
  3.1        192.168.25.31    64: Success  2000: Success  8820: Success
  3.2        192.168.25.32    64: Success  2000: Success  8820: Success
  3.3        192.168.25.33    64: Success  2000: Success  8820: Success
  3.4        192.168.25.34    64: Success  2000: Success  8820: Success
  3.5        192.168.25.55    64: Success  2000: Success  8820: Success
  3.6        192.168.25.56    64: Success  2000: Success  8820: Success

CIFS Credentials: MEDARCH\jqpublic

Capabilities:

  CIFS
    Security Mode  User level, Challenge/response, Signatures disabled
    Server         TCP/445
    Max Request    16644 bytes

Shares:

  CIFS
                                         Storage Space
    Share                              Total (MB)   Free (MB)    Serial Num
    -------------------------------   ----------  ----------    ----------
    ADAM-10                                 17351       11370     5491-7b8b
...
    xand_es_cifs1                           17351       16217     c883-8cc0
    xand_es_cifs2                           17351       16217     c883-8cc0
    -------------------------------   ----------  ----------    ----------
    Share                              Total (MB)   Free (MB)    Serial Num
                                         Storage Space

Time:

  CIFS
    Filer's time is the same as the switch's time.
bstnA6k> ...
```

Showing the Physical Paths for CIFS Shares

For the physical disk and path behind each CIFS share, use the optional paths keyword after the filer hostname/IP:

> **show exports host *filer* paths**
> **[user *username* windows-domain *domain* | proxy-user *proxy*]**
>
> where the options are explained above.

This shows the relationships between shares. If a share is inside the directory tree of another share, it is called a *subshare* of its parent share. You can use this command to identify all share-to-subshare relationships on the filer.

Note

To read the share definitions at the back-end filers, the volume requires Windows credentials with admin-level access. This is a significant increase in access from the standard requirements; choose the user or proxy-user accordingly. If you use a user or proxy-user with lesser permissions, no directory paths appear for the shares.

For example, this shows all CIFS-share paths for the filer at 192.168.25.29. The "CELEBS$," "Y2004," "Y2005" shares are subshares of the "prescriptions" share:

```
bstnA6k> show exports host 192.168.25.29 paths proxy-user acoProxy2
Export probe of filer "192.168.25.29"

CIFS Credentials: MEDARCH\jqpublic

Paths:

  CIFS

    Share                          Directory
    ---------------------------    ------------------------------------
    CELEBS$                        d:\exports\prescriptions\VIP_wing
    Y2004                          d:\exports\prescriptions\2004
    Y2005                          d:\exports\prescriptions\2005
    prescriptions                  d:\exports\prescriptions
bstnA6k> ...
```

Focusing on One Share

To focus on one share, use the share argument in the show exports command:

> **show exports host *filer* share *share-name***
> **[user *username* windows-domain *domain* | proxy-user *proxy*]**

where

> ***share-name*** (1-1024 characters) identifies the share, and
>
> the other options are explained above.

This shows the default report, but only with entries that are relevant to the share. For example, the following command focuses on the ‘histories’ share.

```
bstnA6k> show exports host 192.168.25.20 share histories proxy-user acoProxy2
Export probe of filer "192.168.25.20"

Connectivity:

  Slot.Proc  Proxy Address    Ping (size: result)
  ---------  ---------------  -------------------
  3.1        192.168.25.31    64: Success  2000: Success  8820: Success
  3.2        192.168.25.32    64: Success  2000: Success  8820: Success
  3.3        192.168.25.33    64: Success  2000: Success  8820: Success
  3.4        192.168.25.34    64: Success  2000: Success  8820: Success
  3.5        192.168.25.55    64: Success  2000: Success  8820: Success
  3.6        192.168.25.56    64: Success  2000: Success  8820: Success

CIFS Credentials: MEDARCH\jqpublic

Capabilities:

  CIFS
    Security Mode  User level, Challenge/response, Signatures disabled
    Server         TCP/445
    Max Request    16644 bytes

Shares:

  CIFS
```

```
                                              Storage Space
    Share                               Total (MB)   Free (MB)     Serial Num
    -------------------------------    ----------  ----------     ----------
    histories                                17351       16144      c883-8cc0

Time:

  CIFS
    Filer's time is the same as the switch's time.
bstnA6k> ...
```

Showing Connectivity Only

Use the connectivity keyword to show the Connectivity table alone:

show exports host *filer* [share *share-path*] connectivity

where the options are explained above.

CIFS access is not required for this examination, so you do not need Windows-user credentials.

The ARX sends pings of various sizes from each proxy IP, to confirm that the Maximum Transmission Unit (MTU) is set consistently between each proxy IP and the filer. If any of these pings fails (but not all of them), the filer is unusable because some network equipment between the proxy IP and the filer has an inconsistent MTU setting. (To change the MTU on the ARX, refer to “Enabling Jumbo Frames (optional)” on page 3-6 of the network guide.)

For example, this command shows good connectivity to 192.168.25.20:

```
bstnA6k> show exports host 192.168.25.20 connectivity
Export probe of filer “192.168.25.20”

Connectivity:

  Slot.Proc  Proxy Address    Ping (size: result)
  ---------  ---------------  -------------------
  3.1        192.168.25.31    64: Success  2000: Success  8820: Success
  3.2        192.168.25.32    64: Success  2000: Success  8820: Success
  3.3         192.168.25.33    64: Success  2000: Success  8820: Success
  3.4         192.168.25.34    64: Success  2000: Success  8820: Success
  3.5         192.168.25.55    64: Success  2000: Success  8820: Success
  3.6         192.168.25.56    64: Success  2000: Success  8820: Success
bstnA6k> ...
```

Showing Capabilities Only

The capabilities keyword shows only the Capabilities table:

show exports host *filer* [share *share-path*] capabilities

where the options are explained earlier in the chapter.

As with the connectivity test, this examination does not require a CIFS login. No Windows credentials are required for this filer examination.

For example:

```
bstnA6k> show exports host 192.168.25.20 capabilities
Export probe of filer "192.168.25.20"

CIFS Credentials: [anonymous]

Capabilities:
  NFS
    Port Mapper    TCP/111, UDP/111
    Mount Daemon   V1 TCP/1053, V1 UDP/1053, V2 TCP/1053, V2 UDP/1053, V3 TCP/1053, V3 
UDP/1053
    Server         V2 TCP/2049, V2 UDP/2049, V3 TCP/2049, V3 UDP/2049

  CIFS
    Security Mode  User level, Challenge/response, Signatures disabled
    Server         TCP/445
    Max Request    16644 bytes
bstnA6k> ...
```

Showing Shares Only

To list only the filer’s shares, use the shares keyword:

> **show exports host *filer* shares**
> **[user *username* windows-domain *domain* | proxy-user *proxy*]**
>
> where the options are explained earlier.

The output shows two tables, one for NFS shares and one for CIFS shares. Only the CIFS table appears if you enter Windows credentials.

The NFS table shows each share path and the machines and/or subnets that can access the share. For a share to be eligible for import, all proxy-IP addresses must be on this list.

The CIFS table shows two disk-space measures and the serial number for the storage volume behind the share. If two shares have the same serial number, they are assumed to be shares for the same storage volume on the filer.

For example:

```
bstnA6k> show exports host 192.168.25.20 shares proxy-user acoProxy2
Export probe of filer “192.168.25.20”

CIFS Credentials: MEDARCH\jqpublic

Shares:

  CIFS
                                         Storage Space
    Share                             Total (MB)   Free (MB)    Serial Num
    -------------------------------   ----------  ----------    ----------
...
    histories                              17351       16144     c883-8cc0
...
    xand_es_cifs1                          17351       16220     c883-8cc0
    xand_es_cifs2                          17351       16220     c883-8cc0
    -------------------------------   ----------  ----------    ----------
    Share                             Total (MB)   Free (MB)    Serial Num
```

```
                                          Storage Space
bstnA6k> ...
```

Showing Time Settings

Namespace policy (described in later chapters) requires that the ARX has its clock synchronized with those of its back-end filers. Kerberos authentication also requires synchronized time. You should configure the ARX to use the same NTP servers that the filers use; refer to "Configuring NTP" on page 4-15 in the *CLI Network-Management Guide* for instructions.

To check the clock skew between the ARX and a filer, use the time keyword in the show exports command:

show exports host *filer* [share *share-path*] time

where the options are explained earlier.

This shows the time skew for both NFS and CIFS. No Windows credentials are required to find the CIFS server's clock setting.

For example:

```
bstnA6k> show exports host 192.168.25.20 time
Export probe of filer "192.168.25.20"

CIFS Credentials: [anonymous]

Time:
  NFS

  CIFS
    Filer's time is the same as the switch's time.
bstnA6k> ...
```

Showing CIFS Attributes

This test applies to CIFS shares only. This is not part of the default show exports tests; you must invoke it separately.

Each Windows volume has up to five CIFS attributes that are relevant to namespace imports. These attributes represent support for Compressed Files, Named Streams, Persistent ACLs, Sparse Files, and/or Unicode filenames on disk. This command shows a table of supported CIFS attributes at each of the filer's shares.

show exports host *filer* [share *share-path*] attributes [user *username* windows-domain *domain* | proxy-user *proxy*]

where the options are explained earlier.

For example, this command shows the supported CIFS attributes for the "histories" share on filer 192.168.25.20.

```
bstnA6k> show exports host 192.168.25.20 share histories attributes proxy-user acoProxy2
Export probe of filer "192.168.25.20"

CIFS Credentials: MEDARCH\jqpublic

Attributes:

  CIFS
    Codes: CF=Compressed Files, NS=Named Streams, PA=Persistent ACLs,
           SF=Sparse Files, UD=Unicode On Disk
                                        Attributes
    Share                           CF  NS  PA  SF  UD
    ------------------------------- --  --  --  --  --
    histories                       X   X   X   X   X
bstnA6k> ...
```

Probing for CIFS Security

You can run a probe test to verify that your Windows credentials are adequate for one or more back-end shares. This test applies to CIFS shares only.

This examination attempts to write to the filer's shares and determines whether or not the ***username*** or ***proxy-user*** has Backup Operator privileges at each share. Both Write and Privs should show "OK" status for the share to be eligible for import. If either test fails, try another proxy user, or find a ***username*** that works and add a new proxy user with those credentials (as shown in "Adding a Proxy User" on page 3-2).

This filer examination is more intrusive than the others, so it is not invoked as part of show exports.

From priv-exec mode, use the probe exports command to test some Windows credentials at a given back-end filer:

> **probe exports host *filer* [share *share-path*]
> {user *username* windows-domain *domain* | proxy-user *proxy-user*}**

where the options match those of the show exports command, above.

For example, this command tests the “acoProxy2” credentials at a single back-end share (‘histories’ on filer 192.168.25.20):

```
bstnA6k> enable
bstnA6k# probe exports host 192.168.25.20 share histories proxy-user acoProxy2
Export probe of filer "192.168.25.20"

CIFS Credentials: MEDARCH\jqpublic

Security:

  CIFS

    Description Key: OK (success) NO (failure) -- (not applicable)

    Share                           Write   Privs
    ------------------------------- -----   -----
    histories                       OK      OK
bstnA6k# ...
```

Chapter 6

Adding an External Filer

A Network-Attached Storage (NAS) filer or a file server with Direct-Attached Storage (DAS) is configured as an *external filer* on the ARX. An external filer defines how to access the storage in an external NAS/DAS-based device. From gbl mode, use the external-filer command to create an empty external-filer instance:

> **external-filer *name***
>
> where ***name*** (1-64 characters) is a name you choose for this external filer. This is often associated with the machine's NetBIOS name or hostname, but you can choose any name.

The CLI prompts for confirmation before creating the external-filer object in the database. Enter **yes** to confirm. (You can disable all such create-confirmation prompts with the terminal expert command.)

This puts you into gbl-ext-filer mode. From gbl-ext-filer mode, you need to identify the IP address(es) of the external filer. You can also provide a description and other optional parameters.

For example, the following command sequence creates an external filer named "das1:"

```
bstnA6k(gbl)# external-filer das1
This will create a new filer.


Create filer 'das1'? [yes/no] yes
bstnA6k(gbl-ext-filer[das1])# ...
```

Providing the Filer's IP Address

The next step in external-filer configuration is to give the IP address of the filer. The address must be on the proxy-IP subnet ("Adding a Range of Proxy-IP Addresses" on page 4-6 of the *CLI Network-Management Guide*) or reachable through a gateway on that subnet (via static route: see "Adding a Static Route" on page 4-9 of the same manual). Use the ip address command to identify the external filer:

ip address *ip-address*

where ***ip-address*** is in dotted-decimal format (for example, 192.168.25.19).

For example, the following command sequence declares the external filer, "das1," to be the NAS filer at 192.168.25.19:

```
bstnA6k(gbl)# external-filer das1
bstnA6k(gbl-ext-filer[das1])# ip address 192.168.25.19
bstnA6k(gbl-ext-filer[das1])# ...
```

Providing a Secondary Address (UDP only)

Some filers are configured as redundant pairs that share a single virtual-IP address but occasionally respond to UDP packets from their physical-IP addresses. That is, a client (the ARX) sends a request to the virtual-IP address, and one of the filers responds from its physical-IP address. The switch considers the virtual IP to be *primary*, and the physical IPs to be *secondary*. To identify a secondary IP address, use the secondary flag in the ip address command:

ip address *ip-address* secondary

where ***ip-address*** is a secondary address for the filer.

This only applies to a filer that you will access through UDP; that is, with NFSv2 or NFSv3/UDP. If you only plan to use CIFS and/or NFSv3/TCP, the secondary address is unnecessary.

You can use this flag to configure up to four secondary IP addresses. For example, the following command sequence configures an external filer, "nas1," with one primary address and two secondary addresses:

```
bstnA6k(gbl)# external-filer nas1
bstnA6k(gbl-ext-filer[nas1])# ip address 192.168.25.21
```

```
bstnA6k(gbl-ext-filer[nas1])# ip address 192.168.25.61 secondary
bstnA6k(gbl-ext-filer[nas1])# ip address 192.168.25.62 secondary
bstnA6k(gbl-ext-filer[nas1])# ...
```

Removing a Secondary Address

Use the no form of the command to remove a secondary IP address from the list:

no ip address *ip-address* secondary

where ***ip-address*** is the secondary address to remove.

For example:

```
bstnA6k(gbl)# external-filer nas1
bstnA6k(gbl-ext-filer[nas1])# no ip address 192.168.25.65 secondary
bstnA6k(gbl-ext-filer[nas1])# ...
```

Ignoring a Directory

Some filer shares include read-only directories that should not be aggregated into a namespace, such as backup and snapshot directories. The .snapshot directory, used for backups on some NAS devices, is ignored by default. To exclude another directory, use the ignore-directory command from gbl-ext-filer mode:

ignore-directory *directory*

where ***directory*** (1-256 characters) is the directory to ignore. Do not use any slash (/) characters in this directory name (for example, “dir” is valid, but “dir/subdir” is not).

If the directory name starts with a “.” and is at least three characters long, you can use an asterisk (*) at the end as a wildcard. These wildcards only apply to the root of the share: “.ckpt*” ignores /.ckpt2, but does not ignore /docs/.ckpt7.

The CLI prompts for confirmation if you use a wildcard. Enter **yes** to continue.

You can ignore up to eight directories per external filer. These are common directories to ignore for three filer vendors:

- EMC: .etc, lost+found, .ckpt*

- Network Appliance: .snapshot, ~snapshot
- BlueArc: .snapshot, ~snapshot

Note: Ignore only special, virtual directories designed for filer backups, or directories that only appear in the share's root. If you ignore a standard directory below the root, a client cannot delete the directory's parent.

For example, the following command sequence ignores several directories in the external filer, "nasE1:"

```
bstnA6k(gbl)# external-filer nasE1
bstnA6k(gbl-ext-filer[nasE1])# ignore-directory .etc
bstnA6k(gbl-ext-filer[nasE1])# ignore-directory lost+found
bstnA6k(gbl-ext-filer[nasE1])# ignore-directory .ckpt*
Wildcard matches are only made in the root directory of a share.
Is this the intended behavior? [yes/no] yes
bstnA6k(gbl-ext-filer[nasE1])# ...
```

Re-Instating a Directory

For cases where you want to start using a previously-ignored directory, use the no ignore-directory command from gbl-ext-filer mode:

no ignore-directory *directory*

where ***directory*** (1-256 characters) is the directory to stop ignoring.

For example, the following command sequence to re-instate the ".back" directory for the external filer, "das1:"

```
bstnA6k(gbl)# external-filer das1
bstnA6k(gbl-ext-filer[das1])# no ignore-directory .back
bstnA6k(gbl-ext-filer[das1])# ...
```

Adding a Description (optional)

You can add a description to the filer for use in show commands. The description can differentiate the external filer from others. From gbl-ext-filer mode, use the description command to add a description:

description *text*

where ***text*** is 1-255 characters. Quote the text if it contains any spaces.

For example:

```
bstnA6k(gbl)# external-filer das1
bstnA6k(gbl-ext-filer[das1])# description "shares with financial data (LINUX
filer, rack 14)"
bstnA6k(gbl-ext-filer[das1])# ...
```

Removing the Description

From gbl-ext-filer mode, use no description to remove the description string from the current external-filer configuration:

no description

For example:

```
bstnA6k(gbl)# external-filer fs22
bstnA6k(gbl-ext-filer[fs22])# no description
bstnA6k(gbl-ext-filer[fs22])# ...
```

Setting the CIFS Port (optional)

By default, the ARX sends its CIFS messages to port 445 or 139 at the external filer. Port 445 supports raw CIFS communication over TCP, port 139 supports CIFS through NetBIOS; the ARX tries port 445 first and uses port 139 as a fall-back. For most (if not all) CIFS configurations, this should suffice. For filers that do not listen at either of these well-known ports, use the cifs-port command to set the port:

cifs-port *port*

where ***port*** is a TCP/UDP port number from 1-65535. Numbers from 0-1023 are reserved for "well-known" TCP/UDP ports.

For example, the following command sequence sets the CIFS port to 7231 for the filer named "fs1:"

```
bstnA6k(gbl)# external-filer fs1
bstnA6k(gbl-ext-filer[fs1])# cifs-port 7231
bstnA6k(gbl-ext-filer[fs1])# ...
```

Refer to RFCs 1001 and 1002 for NetBIOS specifications.

Reverting to the CIFS-Port Default

Use no cifs-port to revert back to the default CIFS port, 445 (or 139):

no cifs-port

For example, the following command sequence stops using a pre-set CIFS port for the filer named "fs1:"

```
bstnA6k(gbl)# external-filer fs1
bstnA6k(gbl-ext-filer[fs1])# no cifs-port
bstnA6k(gbl-ext-filer[fs1])# ...
```

Listing External Filers

Use the show external-filer command to get a list of all external filers:

show external-filer

For example, the following command lists all of the external filers known to the ARX:

```
bstnA6k(gbl)# show external-filer
  Name                        IP Address     Description
  ------------------------    -------------  ----------------------------
  das1                       192.168.25.19  financial data (LINUX filer, rack 14)
  fs2                         192.168.25.27  bulk storage server (DAS, Table 3)
  fs1                         192.168.25.20  misc patient records (DAS, Table 3)
  nasE1                       192.168.25.51  NAS filer E1
  fs3                        192.168.25.28  Hematology lab server (DAS, Table 8)
  fs4                         192.168.25.29  prescription records (DAS, Table 3)
  das2                        192.168.25.22  DAS (Solaris) filer 2 (rack 16)
  das3                        192.168.25.23  DAS (Solaris) filer 3 (rack 16)
  nas1                        192.168.25.21  NAS filer 1 (rack 31)
                              192.168.25.61   (secondary)
                              192.168.25.62   (secondary)
  das7                        192.168.25.24  Redhat-LINUX filer 1
  das8                        192.168.25.25  Redhat-LINUX filer 2
  nas2                        192.168.25.44  NAS filer 2 (rack 31)
  nas3                        192.168.25.47  NAS filer 3 (rack 32)
bstnA6k(gbl)# ...
```

Showing External-Filer Details

Identify a particular filer with the show external-filer command to see details about the filer:

show external-filer *filer-name*

where ***filer-name*** (up to 64 characters) is the name given to the external filer.

For example, the following command shows the DAS filer named “das1:”

```
bstnA6k(gbl)# show external-filer das1

Filer “das1” Configuration

  Description         financial data (LINUX filer, rack 14)
  Filer IP            192.168.25.19
  CIFS Port           default (auto-detect)
  NFS TCP Connections 1 (default)

Managed Exports
--------------------------------------------------------------------------------

  NFS Export: /exports/budget
    Namespace: wwmed
    Volume: /acct

  Directories to ignore for importing
  ------------------------------------
    .snapshot

bstnA6k(gbl)# ...
```

Showing Details for all External Filers

Use show external-filer all to see details about every configured external filer:

show external-filer all

Samples - Adding Two Filers

The following command sequence adds two filers to an ARX. The first filer offers three NFS exports and has an IP address of 192.168.25.19. The second filer offers two CIFS shares and has an IP address of 192.168.25.20.

This sequence begins after the filers are connected to the switch through IP configuration:

```
bstnA6k(gbl)# external-filer das1
This will create a new filer.

Create filer 'das1'? [yes/no] yes
bstnA6k(gbl-ext-filer[das1])# ip address 192.168.25.19
```

```
bstnA6k(gbl-ext-filer[das1])# exit
bstnA6k(gbl)#
```

This command sequence creates a new filer, “fs1,” with two CIFS shares:

```
bstnA6k(gbl)# external-filer fs1
This will create a new filer.

Create filer 'fs1'? [yes/no] yes
bstnA6k(gbl-ext-filer[fs1])# ip address 192.168.25.20
bstnA6k(gbl-ext-filer[fs1])# exit
bstnA6k(gbl)#
```

Removing an External Filer

You can remove an external filer from back-end storage by deleting its configuration. To be eligible for deletion, the external filer must not be referenced by any namespace. Use the show external-filer command (see “Showing External-Filer Details” on page 6-7) to see if a namespace is referencing the filer.

Use the no form of the external-filer command to remove an external-filer instance:

no external-filer *nas-name*

where ***nas-name*** (1-64 characters) identifies the external filer.

For example, the following command sequence removes an external filer named “finances:”

```
bstnA6k(gbl)# no external-filer finances
bstnA6k(gbl)# ...
```

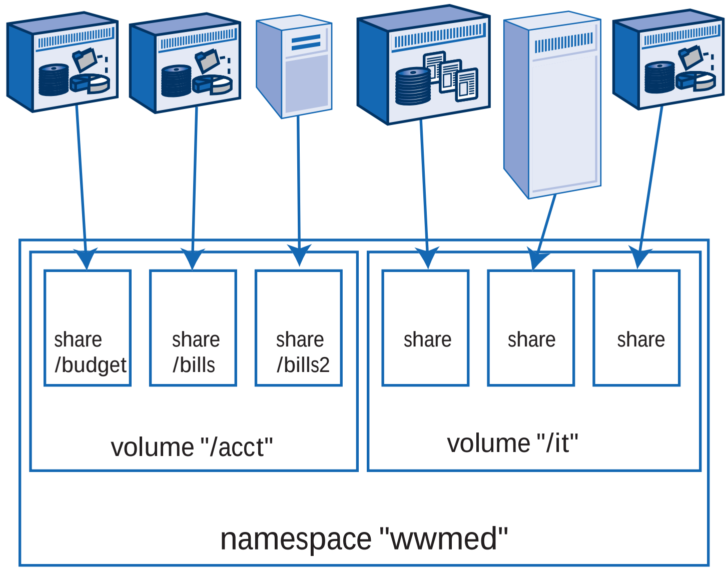
Next

The external (NAS, or DAS-based) filer’s shares are ready to be included in a namespace volume. The next chapters describe how to configure namespaces and various types of volumes.

Chapter 7

Configuring a Namespace

The ARX aggregates storage from external servers into one or more *namespaces*. A namespace is a collection of virtual file systems, called *volumes*. Each volume consists of storage space from any number of Network-Attached Storage (NAS) or filer servers with Direct-Attached Storage (DAS). A volume can contain multiple *shares*, where each share maps to an export or share from an external (NAS/DAS) filer. Clients access the volume as a single directory structure through a single mount point; the volume is analogous to a UNIX partition or a Windows logical drive.



The purpose of the namespace is to contain one or more volumes with a common set of access protocols (CIFS/NFS), authentication mechanisms, and character encoding. This chapter explains how to create a namespace. The next chapters explain how to aggregate your storage into various types of namespace volumes.

From gbl mode, use the namespace command to create a new namespace.

namespace *name*

where ***name*** (1-30 characters) is a name you choose for the namespace. This is only a configuration name. It is not visible to clients.

The CLI prompts for confirmation before creating the namespace; this provides an opportunity to check for mistakes in the name. Enter **yes** to proceed. This puts you into gbl-ns mode, where you configure the volumes for the namespace.

For example, this command set creates a namespace called “wwmed:”

```
bstnA6k(gbl)# namespace wwmed
This will create a new namespace.

Create namespace 'wwmed'? [yes/no] yes
bstnA6k(gbl-ns[wwmed])# ...
```

Concepts and Terminology

A namespace can contain up to three types of volume: managed, direct, or shadow.

A *managed* volume keeps track of all files and directories in the servers behind it. The file and directory locations are called the volume’s *metadata*. A managed volume builds its database of metadata when you first enable it; it scans all of its back-end servers and *imports* all files and directories by recording their locations. By keeping metadata, a managed volume can support policies for balancing or migrating files.

A *direct* volume is a collection of mount points into its back-end storage. It does not import files and directories, does not keep metadata, and does not support policies. The direct volume is useful for quickly aggregating storage into a single mount point.

The *shadow* volume is a frequently-updated duplicate of a managed volume, possibly hosted at a different ARX in the same Resilient-Overlay Network (RON, described in the *CLI Network-Management Guide*).

Listing All Namespaces

To verify a new namespace was created, use the show namespace command to get a list of all namespaces:

show namespace

For example, the following command lists all of the namespaces known to the ARX:

```
bstnA6k# show namespace

Configured Namespaces
---------------------

Namespace                         Description
------------------------------    -----------------------------------------------
medco
wwmed                             namespace for World-Wide Medical network
medarcv
insur                             WW Medical insurance claims and records
bstnA6k# ...
```

Showing Namespace Details

As you configure the namespace, it is useful to periodically view the namespace's current configuration and running state. By including a namespace name, you can use the show namespace command to see details about the namespace and its volumes.

show namespace *name*

where ***name*** (1-30 characters) is the name of the namespace.

In addition to showing the full configuration of the namespace (including a number of components described later in this chapter), this report shows whether or not each namespace share is online.

For example, the following command shows the configuration of the namespace named “wwmed.”

```
bstnA6k# show namespace wwmed

Namespace “wwmed” Configuration
Description namespace for World-Wide Medical network
Metadata Cache Size: 512 MB

Domain Information
------------------



Supported Protocols
-------------------
  nfsv3

Participating Switches
----------------------
  bstnA6k (vpu 1) [Current Switch]



Volumes
-------
  /acct

             Volume freespace: 100GB (automatic)
                Metadata size: 1.5M
          Metadata free space: 32GB
            Import Protection: On
                        State: Enabled

                  Host Switch: bstnA6k
                     Instance: 2
                          VPU: 1 (domain 1)
                        Files: 4.2k used (426 dirs), 3.9M free
```

```
Metadata shares:

  Filer          Backend Path          Contains Metadata  Status
  -------------------------------------------------------------------
  nas1           /vol/vol1/meta1       Yes                Online

Share bills
  Filer                        das8 [192.168.25.25]
  NFS Export                   /work1/accting
  Features                     unix-perm
  Status                       Online
  Critical Share               Yes
  Free space on storage        17GB (18,803,621,888 B)
  Free files on storage        18M
  Transitions                  1
  Last Transition              Wed Apr  4 03:41:05 2007

Share bills2
  Filer                        das3 [192.168.25.23]
  NFS Export                   /data/acct2
  Features                     unix-perm
  Status                       Online
  Free space on storage        4.4GB (4,776,999,936 B)
  Free files on storage        243636
  Transitions                  1
  Last Transition              Wed Apr  4 03:41:06 2007

Share budget
  Filer                        das1 [192.168.25.19]
  NFS Export                   /exports/budget
  Features                     unix-perm
  Status                       Online
  Volume Root Backing          Yes
  Free space on storage        56GB (60,188,938,240 B)
  Free files on storage        7M
```

```
      Transitions                 1
      Last Transition             Wed Apr  4 03:41:04 2007

    Share it5
      Filer                       das7 [192.168.25.24]
      NFS Export                  /lhome/it5
      Features                    unix-perm
      Status                      Online
      Free space on storage       22GB (24,237,064,192 B)
      Free files on storage       12M
      Transitions                 1
      Last Transition             Wed Apr  4 03:41:06 2007

    Share Farms
    -----------
      Share Farm fm1
        Share                     bills
        Share                     bills2
        Share                     budget

        State                     Enabled
        Volume Scan Status        Complete
        Migration Status          Complete
        New File Placement Status Enabled




    Volume Rules
    ---------------
      Rule Name                   docs2das8
        Type                      Place Rule
        State                     Enabled
        Volume Scan Status        Complete
        Migration Status          Complete
        New File Placement Status Enabled

bstnA6k# ...
```

Showing Details for All Namespaces

Use show namespace all to see details about all configured namespaces:

show namespace all

Showing Filer Shares Behind the Namespace

You can use show namespace mapping to list the filer shares that are behind all configured namespaces:

show namespace mapping

For example:

```
bstnA6k# show namespace mapping

Namespace                   Physical Server
---------------------       ---------------------
insur:/claims
                            nas1:/vol/vol1/meta2*
                            \\nas1\insurance
                            \\nasE1\patient_records



Namespace                   Physical Server
---------------------       ---------------------
medarcv:/lab_equipment
                            \\fs2\lab_equipment
                            nas1:/vol/vol1/meta6*

medarcv:/rcrds
                            \\fs1\histories
```

```
                        \\fs1\prescriptions
                        \\fs2\bulkstorage
                        nas1:/vol/vol1/meta3*

medarcv:/test_results
  chemLab               \\fs1\chem_results/.
  hematologyLab         \\fs3\hematology_results/.



Namespace               Physical Server
--------------------    ---------------------
medco:/vol
  vol0/corp             nas1:/vol/vol0/direct/shr
  vol0/notes            nas1:/vol/vol0/direct/notes
  vol1/mtgMinutes       nas2:/vol/vol1/direct/mtgs
  vol1/sales            nas2:/vol/vol1/direct/export
  vol2                  nas3:/vol/vol2/direct/data



Namespace               Physical Server
--------------------    ---------------------
wwmed:/acct
                        das1:/exports/budget
                        das3:/data/acct2
                        das7:/lhome/it5
                        das8:/work1/accting
                        nas1:/vol/vol1/meta1*



Where * denotes metadata only physical server.
bstnA6k# ...
```

Showing Shares Behind One Namespace

Add a namespace name to show only the shares behind that particular namespace:

show namespace mapping *name*

where ***name*** (1-30 characters) is the name of the namespace.

For example, this shows the filer shares behind the “wwmed” namespace:

```
bstnA6k# show namespace mapping wwmed

Namespace                  Physical Server
---------------------      ----------------------
wwmed:/acct
                           das1:/exports/budget
                           das3:/data/acct2
                           das7:/lhome/it5
                           das8:/work1/accting
                           nas1:/vol/vol1/meta1*


Where * denotes metadata only physical server.
bstnA6k# ...
```

Setting the Namespace Protocol(s)

The next step in creating a namespace is declaring the protocol(s) clients must use to access its data. All of the external filers in the namespace must support this entire protocol set; if the namespace supports both NFSv2 and NFSv3, all of its filer shares must also support both of those versions of NFS. Use the gbl-ns protocol command to identify one protocol for the namespace:

protocol {nfs2 | nfs3 | nfs3tcp | cifs}

where you must choose one of the flags, **nfs2**, **nfs3** (over UDP), **nfs3tcp** (NFSv3 over TCP) or **cifs**; these specify the protocol(s) supported by all of the namespace’s filers. Repeat the command to specify multiple protocols for the namespace.

For example, this command set allows two forms of NFS access to the “wwmed” namespace:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# protocol nfs2
bstnA6k(gbl-ns[wwmed])# protocol nfs3
bstnA6k(gbl-ns[wwmed])# ...
```

Removing a Protocol

Use the no form of the protocol command to remove a protocol from the namespace.

no protocol {nfs2 | nfs3 | nfs3tcp | cifs}

where you must choose one of the flags, **nfs2**, **nfs3** (over UDP), **nfs3tcp** (NFSv3 over TCP) or **cifs**.

For example, this command set removes NFSv2 from the “wwmed” namespace:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# no protocol nfs2
bstnA6k(gbl-ns[wwmed])# ...
```

Creating a Multi-Protocol (NFS and CIFS) Namespace

If all the back-end shares support both NFS and CIFS, you can configure a *multi-protocol namespace*. Clients can access the same files from either a CIFS or an NFS client.

The namespace can be backed by a heterogeneous mix of multi-protocol filers, possibly from multiple vendors. The switch passes client requests to these filers, and passes filer responses back to the clients. File attributes, such as file ownership and permission settings, are managed by each filer. Each filer also manages its file and directory naming; if a name is legal in NFS but illegal in CIFS, each filer creates a filer-generated name (FGN) for its CIFS clients. Different vendors use different conventions for attribute conversions and FGNs, so that a CIFS-side name and/or ACLs at one filer may be different at another filer.

These issues are important to managed-volume configuration, and are discussed at length in the managed-volume chapter (as well as the *CLI Maintenance Guide*).

Changing Protocols After Import

We strongly recommend that you choose your protocol set carefully, before configuring any volumes in the namespace. After a managed volume and at least one of its shares is enabled (as described later in the managed-volume chapter), the managed volume imports files and directories from its enabled shares. Protocol changes after this first import require greater care, since they may affect client access to the volume. The following changes affect client service:

- converting a single-protocol namespace to a multi-protocol namespace,
- removing CIFS, or
- removing all support for one version of the NFS protocol (that is, removing NFS2 or removing both of NFS3 and NFS3/TCP).

To make these conversions, you take the managed volume offline (with NSCK destage, described in the *CLI Maintenance Guide*), change the protocol with this command, and restart the imports by re-enabling the managed volume.

For other protocol changes, such as adding NFSv2 to a running NFSv3 namespace, you must disable the volume (using `no enable`) and its front-end NFS service (described in a later chapter), change the protocol, then re-enable both. This causes a shorter service outage.

Setting NFS Character Encoding

NFS *character encoding* refers to the encoding used for each character in an NFS filename. This is the mapping of binary numbers to character symbols. ASCII is an example of a character encoding used commonly in the United States.

A site has a character encoding for NFS filenames that is shared amongst all clients and servers. This should be well-established before the installation of the ARX.

In a managed volume, a multi-protocol namespace prevents CIFS clients from creating a filename that cannot translate directly into the NFS character encoding. For example, if the NFS encoding is ISO 8859–1 (Latin1, a single-byte encoding) and a CIFS user tries to create a file with Korean (multi-byte) characters in the name, the namespace rejects the create operation. A multi-protocol namespace only allows characters that are mappable between NFS and CIFS.

Improper encoding can also present problems during managed-volume import. A file with a non-mappable CIFS character is imported using its NFS-side name; this may not have any resemblance to the intended CIFS-side name. A directory with an non-mappable character can be renamed during import to preserve its resemblance with the original CIFS-side name. Each share generates an import report containing the original CIFS-side name and the new name for each renamed file or directory.

From gbl-ns mode, use the character-encoding nfs command to set the character encoding for NFS filenames:

character-encoding nfs {iso–8859–1 | utf–8}

where:

iso–8859–1 is ISO 8859–1 (Latin1, single-byte) character encoding, and

utf–8 specifies UTF–8 (Unicode) character encoding.

The default is ISO 8859–1 (Latin1).

For example, this command sequence sets character encoding to “UTF-8” (Unicode) for NFS filenames in the ‘insur’ multi-protocol namespace:

```
bstnA6k(gbl)# namespace insur
bstnA6k(gbl-ns[insur])# protocol cifs
bstnA6k(gbl-ns[insur])# protocol nfs3
bstnA6k(gbl-ns[insur])# protocol nfs3tcp
bstnA6k(gbl-ns[insur])# character-encoding nfs utf-8
bstnA6k(gbl-ns[insur])# ...
```

Presumably, all of the multi-protocol filers behind the ‘insur’ namespace also support UTF-8 for their NFS filenames.

Setting CIFS Character Encoding

When a volume from a CIFS namespace is exported through a virtual server (described in a later chapter), the virtual server may register its NetBIOS name with a WINS server. Use the character-encoding cifs command to set the character encoding expected by the local WINS server:

character-encoding cifs {iso–8859–1 | utf–8 | shift-jis | ksc5601}

where

iso–8859–1 is ISO 8859–1 (Latin1, single-byte) character encoding,

utf–8 specifies UTF–8 (Unicode) character encoding,

shift-jis (Microsoft Shift-JIS) supports Japanese characters, and

ksc5601 supports Korean characters.

As with NFS, the default is ISO 8859–1 (Latin1).

For example, this command sequence sets character encoding to “Shift JIS” (Japanese) for CIFS NETBIOS names for the ‘insur’ multi-protocol namespace:

```
bstnA6k(gbl)# namespace insur
bstnA6k(gbl-ns[insur])# character-encoding cifs shift-jis
bstnA6k(gbl-ns[insur])# ...
```

Returning to Default Character Encoding

Use no character-encoding to revert to ISO 8859–1 for either NFS filenames or CIFS NetBIOS names:

no character-encoding {nfs | cifs}

For example:

```
bstnA6k(gbl)# namespace insur
bstnA6k(gbl-ns[insur])# no character-encoding cifs
bstnA6k(gbl-ns[insur])# ...
```

You cannot change the character encoding after any of the namespace's managed volumes are enabled, as described in a later chapter.

Configuring Windows Authentication (CIFS)

This section applies only to a namespace that supports CIFS. Skip to the next section if this is an NFS-only namespace.

To configure Windows NTLM or Kerberos authentication for the namespace, you first declare the proxy user for the namespace. A *proxy user* configuration contains a Windows domain, username, and password. The namespace software uses these credentials for authenticating with back-end CIFS shares. This is for operations that do not directly involve a Windows client: initial inventory of the CIFS shares and migration of files from one share to another. (Migration is a method for enforcing several namespace policies, which are described in later chapters.)

You must configure a proxy user for the namespace's domain ahead of time: the steps were described earlier in "Adding a Proxy User" on page 3-2. The proxy user *must* belong to the Backup Operators group for all CIFS filers in the namespace. You can use the proxy user in probe exports to test this; this was described in "Probing for CIFS Security" on page 5-14.

Each namespace can have one proxy user. The proxy-user credentials must be valid in the same Windows domain as the namespace. Multiple namespaces in the same domain can use the same proxy user. From gbl-ns mode, use the proxy-user command to apply a proxy user to the namespace:

proxy-user *name*

where ***name*** (1-32 characters) identifies the proxy user for this namespace. Use the show proxy-user command for a list of configured proxy users.

For example, this command set applies a proxy user, "acoProxy2," to the "medarcv" namespace:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# show proxy-user

Name                                Domain             User
-------------------------------------------------------------------------------
acoProxy1                           WWMEDNET           jqprivate
```

```
acoProxy3                         FDTESTNET        jqtester
acoProxy2                         MEDARCH          jqpublic
bstnA6k(gbl-ns[medarcv])# proxy-user acoProxy2
bstnA6k(gbl-ns[medarcv])# ...
```

Using Kerberos for Client Authentication

You can configure the namespace to authenticate its clients with Kerberos instead of (or in addition to) NTLM. If you plan to use NTLM only, skip ahead to the next section.

From gbl-ns mode, use the kerberos-auth command to enable Kerberos authentication for the namespace's clients:

kerberos-auth

You can only add or remove Kerberos authentication while the namespace is disabled. A later section explains how to enable and disable a namespace.

For example:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# kerberos-auth
bstnA6k(gbl-ns[medarcv])# ...
```

Disabling Kerberos

Use the no kerberos-auth command to disable Kerberos authentication:

no kerberos-auth

For example:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# no kerberos-auth
bstnA6k(gbl-ns[medarcv])# ...
```

Identifying the NTLM Authentication Server

NTLM authentication also requires a mechanism for authenticating Windows clients at the namespace's back-end filers. Kerberos-only sites do not require any NTLM configuration, though you can configure a namespace that supports both authentication protocols.

An *NTLM-authentication server* is a Windows Domain Controller (DC) that is the host for an Acopia Secure Agent (ASA). The namespace software uses the NTLM-authentication server to authenticate its clients. The ASA makes it possible for the namespace to act as a proxy, re-using the client's credentials at multiple back-end shares. Each ASA provides authentication for a single Windows domain.

You separately install an ASA at its DC, then you specify the server's IP address (and other parameters) at the ARX's CLI. Refer to the *Secure Agent Installation Guide* for instructions to install an ASA and then configure it at the ARX.

Once the NTLM-authentication server is configured, you can use the server with one or more namespaces. From gbl-ns mode, use the ntlm-auth-server command to apply a server to the namespace:

ntlm-auth-server *name*

where ***name*** identifies the NTLM authentication server for this namespace. Use show ntlm-auth-server for a list of them; recall "Listing NTLM Authentication Servers" on page 3-7.

As with Kerberos, you can only add or remove NTLM authentication while the namespace is disabled.

For example, this command set finds the "dc1" authentication server, shows the server configuration, then applies the authentication server to the "medarcv" namespace:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# show ntlm-auth-server


Name
----------------------------------------
dc1


bstnA6k(gbl-ns[medarcv])# show ntlm-auth-server dc1
```

```
Name                Domain Name       Server           Port
-------------------------------------------------------------------------------
dc1                 MEDARCH           192.168.25.102   25805

Mapped to the Following Namespaces
-------------------------------------------------------------------------------
insur
bstnA6k(gbl-ns[medarcv])# ntlm-auth-server dc1
bstnA6k(gbl-ns[medarcv])# ...
```

Multi-Domain Support

You can use the ntlm-auth-server command multiple times to support multiple Windows Domains. Each NTLM authentication server supports a single Windows domain; by identifying an NTLM authentication server you support all clients from that Windows Domain. For example, if you identify a server in MEDARCH and another server in the LOWELL domain, clients from both MEDARCH and LOWELL can access the namespace:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# ntlm-auth-server dc1
bstnA6k(gbl-ns[medarcv])# ntlm-auth-server dc-lowell
bstnA6k(gbl-ns[medarcv])# ...
```

Removing an NTLM-Authentication Server

From gbl-ns mode, use no ntlm-auth-server to remove a server from the namespace:

no ntlm-auth-server *name*

where ***name*** identifies the NTLM authentication server to remove from the namespace.

Caution

If you remove an NTLM-authentication server from the namespace, the server's clients will no longer be able to authenticate through NTLM. This may even pose a problem for Kerberos clients who occasionally fall back to NTLM authentication.

While the namespace is enabled, you cannot remove the final NTLM-authentication server.

For example, this command set removes the "asa-test" authentication server from the "medarcv" namespace:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# no ntlm-auth-server asa-test
bstnA6k(gbl-ns[medarcv])# ...
```

Opening Windows-Management Access (optional, MMC)

You can configure a group of Windows clients to have management authorization in this namespace, typically through the Microsoft Management Console (MMC). These clients can use MMC to make CIFS shares, open files, and/or any open CIFS-client sessions. You configure this group of clients and their permissions ahead of time, as described in "Authorizing Windows-Management (MMC) Access" on page 3-28.

Each namespace can allow multiple management-authorization groups. From gbl-ns mode, use the windows-mgmt-auth command to apply one such group to the namespace:

windows-mgmt-auth *name*

where ***name*** (1-64 characters) identifies one management-authorization group for this namespace. Use the show windows-mgmt-auth command for a list of all configured management-authorization groups.

Enter this command once for each authorized group.

For example, this command set applies three management-authorization groups to the “medarcv” namespace:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# show windows-mgmt-auth
...
bstnA6k(gbl-ns[medarcv])# windows-mgmt-auth testGroup
bstnA6k(gbl-ns[medarcv])# windows-mgmt-auth fullAccess
bstnA6k(gbl-ns[medarcv])# windows-mgmt-auth readOnly
bstnA6k(gbl-ns[medarcv])# ...
```

Removing a Management-Authorization Group

Use no windows-mgmt-auth to remove one management-authorization group from the current namespace:

no windows-mgmt-auth *name*

where ***name*** (1-64 characters) identifies one management-authorization group to remove from the namespace.

For example, this command sequence deletes the “testGroup” from the “medarcv” namespace:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# no windows-mgmt-auth testGroup
bstnA6k(gbl-ns[medarcv])# ...
```

Selecting a SAM-Reference Filer

CIFS clients, given sufficient permissions, can change the users and/or groups who have access to a given file. For example, the owner of the “penicillin.xls” file can possibly add “nurses” or “doctors” to the list of groups with write permission. The list of groups in the network is traditionally provided by the Security Account Management (SAM) database on the file’s server. The ARX does not provide an internal SAM database; the namespace proxies all SAM queries to one of its CIFS filers, chosen at random.

Note

According to the CIFS protocol, a CIFS-client application sends all of its SAM queries through a special pseudo share, IPC$. Each CIFS namespace therefore has a separate pseudo volume that it shares as IPC$. Since the queries come to this volume instead of the file’s volume, the namespace software does not know the file’s volume. Therefore, the namespace cannot intelligently choose an appropriate back-end filer as its SAM reference.

The filer used for SAM queries must contain a super set of all groups in all volumes, or some of the groups will be missing from the list. If any volume has filers that support Local Groups (as opposed to groups defined at the DC), you must configure one filer with all groups. If none of the filers use local groups, you can skip to the next section; the namespace can choose any of the filers as its SAM reference.

Use the gbl-ns sam-reference command to identify the pre-configured filer:

sam-reference *filer*

where ***filer*** (1-64 characters) is the external filer’s name, as displayed in show external-filer (see “Listing External Filers” on page 6-6).

For example, the following command sequence uses the “fs2” filer as a SAM reference for all volumes in the medarcv namespace:

```
bstnA6k(gbl)# show external-filer
  Name                           IP Address     Description
  -------------------------    -------------  ----------------------------
  das1                        192.168.25.19  financial data (LINUX filer, rack 14)
  fs2                           192.168.25.27  bulk storage server (DAS, Table 3)
  fs1                           192.168.25.20  misc patient records (DAS, Table 3)
  nasE1                         192.168.25.51  NAS filer E1
  fs3                          192.168.25.28  Hematology lab server (DAS, Table 8)
  fs4                           192.168.25.29  prescription records (DAS, Table 3)
  das2                          192.168.25.22  DAS (Solaris) filer 2 (rack 16)
  das3                          192.168.25.23  DAS (Solaris) filer 3 (rack 16)
  nas1                          192.168.25.21  NAS filer 1 (rack 31)
                                192.168.25.61   (secondary)
                                192.168.25.62   (secondary)
  das7                          192.168.25.24  Redhat-LINUX filer 1
  das8                          192.168.25.25  Redhat-LINUX filer 2
  nas2                          192.168.25.44  NAS filer 2 (rack 31)
  nas3                          192.168.25.47  NAS filer 3 (rack 32)
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# sam-reference fs2
bstnA6k(gbl-ns[medarcv])# ...
```

Adding a Volume

The next, most important step in creating a namespace is creating one or more *volumes* for it. Each volume in a namespace is like a single file system. The volume aggregates one or more exports/shares from actual filers.

There are three forms of namespace volume: managed, direct, and shadow. The default type is a *managed volume*, so-called because it keeps metadata on all of the shares behind it and uses the metadata to manage the shares. A *direct volume* (called a *presentation volume* in the GUI) does not have any metadata or support policy. A *shadow volume* is a read-only copy of one or more managed volumes.

The next two chapters provide detailed instructions for configuring direct and managed volumes. Shadow volumes are described in a chapter after the storage-policy chapters. One namespace can support all three types of volumes.

From gbl-ns mode, use the volume command to create a volume:

volume *topdir*

where ***topdir*** (1-1024 characters) is the directory with the contents of the filer shares. Volumes cannot contain one another: if you make a “/var” volume, you cannot have a “/var/log” volume in the same namespace; if you make a “/” volume, you cannot have any other volumes.

For a new volume, the CLI prompts for confirmation before adding it to the namespace. Enter **yes** to proceed. This puts you into gbl-ns-vol mode, where you must declare at least one share.

For example, this command set creates a single volume ("/acct") for the "wwmed" namespace:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
This will create a new volume.

Create volume '/acct'? [yes/no] yes
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Volume configuration is the bulk of namespace configuration. As mentioned above, the next chapters explain how to configure and maintain each volume type.

Enabling the Namespace (optional)

The final step in configuring a namespace is to enable it. This is optional, as each enabled volume in the namespace can process client requests independently. This command serially enables every volume in the namespace.

From gbl-ns mode, use the enable command to enable the current namespace:

enable

If the namespace contains any managed volumes, this causes all of them to start importing their enabled shares.

For example, the following command sequence enables the namespace named "wwmed:"

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# enable
bstnA6k(gbl-ns[wwmed])# ...
```

All imports happen asynchronously, so that you can issue more CLI commands while the imports happen in the background. You can use show namespace [status] to check the progress of the imports; this is discussed later in the managed-volume chapter.

Enabling All Shares in the Namespace

From gbl-ns mode, you can enable all of the namespace's shares with a single command. Use the enable shares command to accomplish this:

enable shares

For example, the following command sequence enables all shares in the "wwmed" namespace:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# enable shares
bstnA6k(gbl-ns[wwmed])# ...
```

Taking Ownership of All Managed Shares (optional)

Before a managed volume imports each share, it checks the root directory in the share for a special file that marks it as "owned" by an ARX. If this marker file exists, the managed volume does not proceed with the import; no two volumes can manage the same share. You may need to override this safety mechanism for a special case.

Consider an installation that uses legacy, filer-based applications to prepare for disaster recovery: it copies all of its directories and files from a primary site to another site. If an ARX manages the directories at the primary site, it places its ownership marker in the root of each share. The filer-based application copies the marker files to the remote site, along with all data files. An ARX at the backup site cannot import these shares because of the markers.

This does not apply to any direct volumes in the namespace.

You can use the optional take-ownership flag for this special case. If any managed volume finds an ownership marker in the root of a share, it overwrites the marker file. Otherwise, it imports the share as usual:

enable shares take-ownership

Caution

Do not use this option if it is possible that another ARX is managing one of the namespace's shares. This would unexpectedly remove the share(s) from service at the other ARX.

The CLI prompts for confirmation before taking ownership of any shares. Enter **yes** to proceed.

For example, the following command sequence enables all shares in the “insur_bkup” namespace and, if necessary, takes ownership of all of them:

```
prtlndA1k(gbl)# namespace insur_bkup
prtlndA1k(gbl-ns[insur_bkup])# enable shares take-ownership
This command allows the switch to virtualize shares that are used by other Acopia switches.
Allow switch to take ownership of all the shares in this namespace? [yes/no] yes
prtlndA1k(gbl-ns[insur_bkup])# ...
```

Disabling All Shares

Use no enable shares command to disable each of the namespace’s individual shares:

no enable shares

For example, this command sequence disables all of the shares in the “ns1” namespace:

```
bstnA6k(gbl)# namespace ns1
bstnA6k(gbl-ns[ns1])# no enable shares
bstnA6k(gbl-ns[ns1])# ...
```

Disabling the Namespace

You can disable a namespace to stop clients from accessing it. This disables every volume in the namespace. Use no enable in gbl-ns mode to disable the namespace:

no enable

For example, the following command sequence disables the “wwmed” namespace:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# no enable
bstnA6k(gbl-ns[wwmed])# ...
```

Showing Namespace Configuration

To review the configuration settings for a namespace, use the show global-config namespace command:

> **show global-config namespace [*name*]**
>
> where ***name*** (optional, 1-30 characters) identifies the namespace. If you omit this, the output includes all namespaces

The output shows all of the configuration options required to recreate the namespace. The options are in order, so that they can be used as a CLI script.

For example, the following command shows the configuration for the “wwmed” namespace:

```
bstnA6k# show global-config namespace wwmed
;=============================== namespace ===============================
namespace wwmed
  description “namespace for World-Wide Medical network”
  protocol nfs3
  volume /acct
    import protection
    metadata critical
    modify
    policy pause backupWindow
    reimport-modify
    reserve files 4000000
    hosted-by bstnA6k
    metadata share nas1 nfs3 /vol/vol1/meta1
    share bills
      critical
      filer das8 nfs /work1/accting
      enable
      exit

    share bills2
      filer das3 nfs /data/acct2
      enable
      exit

    share budget
      filer das1 nfs /exports/budget
      enable
      exit

    share it5
      filer das7 nfs /lhome/it5
      enable
      exit

    share-farm fm1
      share budget
      share bills
      share bills2
      maintain-free-space 2G
      auto-migrate 2G
      balance Capacity
      enable
      exit

    place-rule docs2das8
      report docsPlc verbose
      from fileset bulky
      target share bills
      limit-migrate 50G
      enable
      exit

    vpu 1 domain 1
    enable
    exit
```

```
  exit

bstnA6k# ...
```

Removing a Namespace

From priv-exec mode, you can use the remove namespace command to remove a namespace and all of its volumes:

remove namespace *name* [timeout *seconds*] [sync]

where:

name (1-30 characters) is the namespace to remove,

seconds (optional, 300-10,000) sets a time limit on each of the removal's component operations, and

sync (optional) waits for the removal to finish before returning. With this option, the CLI lists the namespace components as it removes them.

The CLI prompts for confirmation before removing the namespace. Enter **yes** to continue.

This operation generates a report, "removeNs_*namespace*_*date*.rpt," which catalogs all of the actions that it took. The *namespace* in the filename identifies the removed namespace, and the *date* is the date and time when the command started. The CLI shows the report name after you invoke the command. Use show reports to see the file listing; use show, tail, or grep to read the file. To save the report off to an external site, use the copy command from priv-exec mode.

The command does not create the report if you use the sync option; it shows its actions at the command line instead.

For example, this command sequence exits to priv-exec mode and then synchronously removes the insur_bkup namespace:

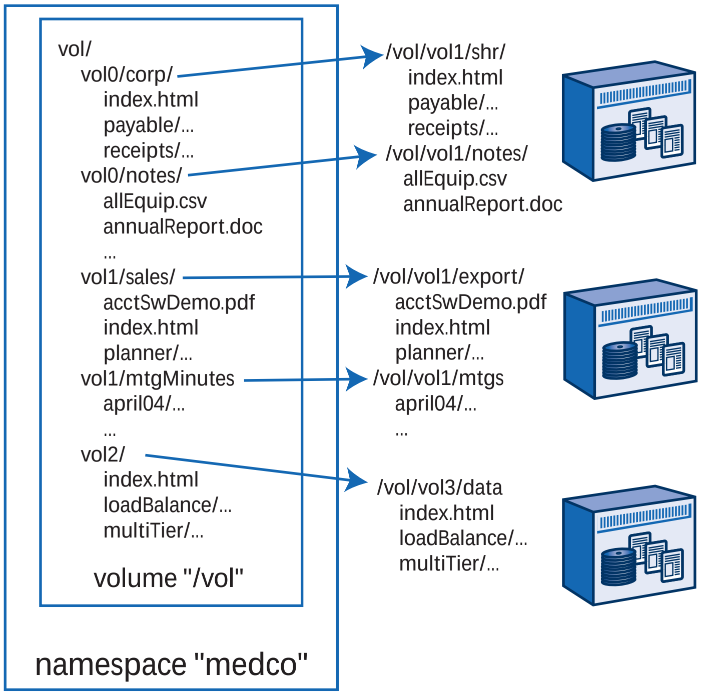
```
prtlndA1k(gbl)# end
prtlndA1k# remove namespace insur_bkup sync

Remove namespace 'insur_bkup'? [yes/no] yes
```

```
% INFO: Removing service configuration for namespace insur_bkup
% INFO: Removing CIFS browsing for namespace insur_bkup
% INFO: Removing volume policies for namespace insur_bkup
% INFO: destroy policy insur_bkup /insurShdw
% INFO: Removing shares for namespace insur_bkup
% INFO: no share backInsur
% INFO: Removing volume metadata shares for namespace insur_bkup
% INFO: no metadata share nas-p1 path /vol/vol1/mdata_B
% INFO: Removing volumes for namespace insur_bkup
% INFO: Removing NFS services for namespace insur_bkup
% INFO: Removing CIFS services for namespace insur_bkup
% INFO: no volume /insurShdw
% INFO: Removing namespace metadata shares for namespace insur_bkup
% INFO: Removing namespace insur_bkup
% INFO: no namespace insur_bkup
prtlndA1k# ...
```

Chapter 8

Adding a Direct Volume

Each share in a *direct volume* attaches one or more of its own virtual directories to real directories at back-end shares. These *attach points* are analogous to mount points in NFS and network-drive connections in CIFS. The back-end directory trees are all reachable from the same volume root.The volume does not record the contents of the back-end shares, so it does not keep metadata or support storage policies. This is called a *presentation volume* in the GUI.



A direct volume is easier to set up than a managed volume, so this chapter is offered before the managed-volume chapter.

As explained earlier (in “Adding a Volume” on page 7-21), you use the gbl-ns volume command to create a volume. This puts you into gbl-ns-vol mode, where you must declare this volume for use as a direct volume and create at least one direct share.

Note

Direct volumes do not support NFSv2. This protocol is set in the volume’s namespace, as described earlier in “Setting the Namespace Protocol(s)” on page 7-9.

For example, this command set creates a single volume (“/vol”) for the “medco” namespace:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
This will create a new volume.

Create volume '/vol'? [yes/no] yes
bstnA6k(gbl-ns-vol[medco~/vol])# ...
```

Declaring the Volume “Direct”

A new volume is a managed volume by default; the next chapter describes the more-complex managed volumes. From gbl-ns-vol mode, use the direct command to convert a managed volume into a direct volume:

direct

For example, this command sequence converts the volume, “medco~/vol,” into a direct volume:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# direct
bstnA6k(gbl-ns-vol[medco~/vol])# ...
```

Reverting to a Managed Volume

If a direct volume has no attach points configured, you can use no direct to revert the volume back to a managed volume:

no direct

For example, this command sequence ensures that “wwmed~/acct” is a managed volume:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# no direct
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Manually Setting the Volume's Free Space (optional)

The next step in creating a volume is to choose an algorithm for calculating its free space. This is the free-space calculation that is passed onto the client: whenever a user mounts a volume (NFS) or maps a network drive to it (CIFS), this total is the free space that they see.

By default, the namespace volume's free space is the sum of the free space on all of its back-end shares *except* shares from the same storage volume. If two or more shares report the same ID for their backing volume, only the first share is counted. (For NFS shares, the volume uses the file system ID (FSID); for CIFS shares, it uses the Volume Serial Number). If this default is acceptable, you can skip this section.

You may wish to manually control the free space calculation on a share-by-share basis. This means counting all free space on all shares, regardless of duplicate back-end-volume IDs, then ignoring certain shares manually or adjusting their freespace reporting. You can use the freespace ignore and freespace adjust commands, described later, to ignore the freespace from a share or change the reported value for the freespace.

From gbl-ns-vol mode, use the freespace calculation manual command to override the default free-space calculation:

freespace calculation manual

You can set this any time, even after the volume is enabled.

For example, this command sequence makes the ‘access~/G’ volume count the free space in all back-end shares, even multiple shares that draw from the same back-end storage:

```
prtlndA1k(gbl)# namespace access
prtlndA1k(gbl-ns[access])# volume /G
prtlndA1k(gbl-ns-vol[access~/G])# freespace calculation manual
prtlndA1k(gbl-ns-vol[access~/G])# ...
```

Using Automatic Free-Space Calculation

By default, free space is calculated based on the IDs of the volumes behind the back-end shares. If any of these shares report the same volume ID, their free space is counted only once. To return to this default, use no freespace calculation manual:

no freespace calculation manual

For example:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# no freespace calculation manual
bstnA6k(gbl-ns-vol[medco~/vol])# ...
```

Setting CIFS Options

The next step in configuring a volume is addressing its CIFS options, if necessary: This section does not apply to volumes in NFS-only namespaces; skip to the next section if this namespace does not support CIFS.

There are five CIFS-volume attributes that back-end filers may or may not support. They are named streams, compressed files, persistent ACLs, Unicode file names on disk, and sparse files. Each volume can support any and all of these capabilities. However, *all* filer shares used in a volume *must* support the capabilities advertised in the volume. For example: if your volume is going to support compressed files, then *all* of its back-end filer shares must also support compressed files.

By default, new volumes conform to the CIFS options at the first-enabled share. If you try to enable another volume share that does not support one or more of those options, the you get an error and the share remains disabled. You can then disable the unsupported options (the error message tells you which ones) and retry the enable. For an earlier view of all CIFS options on the filer, you can use the show exports command; refer back to "Showing CIFS Attributes" on page 5-13.

From gbl-ns-vol mode, use any combination of the following gbl-ns-vol commands to manually set the options:

[no] named-streams

[no] compressed-files

[no] persistent-acls

[no] unicode-on-disk

[no] sparse-files

For example, the following command sequence disables named streams and sparse files in the "medarcv~/test_results" volume:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /test_results
bstnA6k(gbl-ns-vol[medarcv~/test_results])# no named-streams
bstnA6k(gbl-ns-vol[medarcv~/test_results])# no sparse-files
bstnA6k(gbl-ns-vol[medarcv~/test_results])# ...
```

As another example, this command sequence reinstates named streams in the same volume:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /test_results
bstnA6k(gbl-ns-vol[medarcv~/test_results])# named-streams
bstnA6k(gbl-ns-vol[medarcv~/test_results])# ...
```

Disabling CIFS Oplocks (optional)

The CIFS protocol supports *opportunistic locks* (oplocks) for its files. A client application has the option to take an oplock as it opens a file. While it holds the oplock, it can write to the file (or a cached copy of the file) knowing that no other CIFS client can write to the same file. Once another client tries to access the file for writes, the server gives the first client the opportunity to finish writing. This feature makes it possible for clients to cache their file-writes locally, thereby improving client performance.

CIFS volumes provide oplock support by default. Some installations prefer not to offer oplocks to their CIFS clients. A volume with oplocks disabled does not grant any oplocks to any CIFS clients, so CIFS clients cannot safely use file caching as described above. From gbl-ns-vol mode, use the cifs oplocks-disable command to disable oplock support in the current volume:

cifs oplocks-disable

For example, the following command sequence disables oplock support in “access~/G:”

```
prtlndA1k(gbl)# namespace access
prtlndA1k(gbl-ns[access])# volume /G
prtlndA1k(gbl-ns-vol[access~/G])# cifs oplocks-disable
prtlndA1k(gbl-ns-vol[access~/G])# ...
```

Allowing the Volume to Automatically Disable Oplocks

You can configure the volume to automatically disable oplocks for a CIFS client that times out in response to an “oplock break” command. The “oplock break” command informs a client that it must finish its writes and release the oplock, so that another client can write to the file. For each client that does not respond in 10 seconds or less, the volume disables oplocks for 10 minutes, then re-enables oplocks for that client and tries again. The volume manages oplocks on a client-by-client basis.

To permit the volume to disable oplocks for misbehaving clients, use the optional auto flag in the cifs oplocks-disable command:

cifs oplocks-disable auto

For example, the following command sequence changes the “medarcv~/test_results” volume to automatically disable oplocks:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /test_results
bstnA6k(gbl-ns-vol[medarcv~/test_results])# cifs oplocks-disable auto
bstnA6k(gbl-ns-vol[medarcv~/test_results])# ...
```

Reinstating Oplock Support

Use no cifs oplocks-disable to support CIFS oplocks in the current volume:

no cifs oplocks-disable

For example, the following command sequence enables oplocks in the “medco~/vol” volume:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# no cifs oplocks-disable
bstnA6k(gbl-ns-vol[medco~/vol])# ...
```

Adding a Share

The next step in creating a direct volume is identifying one or more shares for it. A *share* maps to a single CIFS share or NFS export from an external (NAS or DAS) filer. A *direct share* contains one or more attach-point directories, each attached to a real directory on its back-end share. A direct volume can contain dozens of shares, where each could possibly correspond to a different filer.

From gbl-ns-vol mode, use the share command to add a share to a volume:

share *name*

where ***name*** (1-64 characters) is a name you choose for the new share.

As with new namespaces and volumes, the CLI prompts for confirmation before creating a new share. Enter **yes** to proceed. This puts you into gbl-ns-vol-shr mode, where you must identify the filer and export/share, and then you must enable the namespace share.

For example, this command set adds a share called “corporate” to the “/vol” volume in the “medco” namespace:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# share corporate
This will create a new share.

Create share 'corporate'? [yes/no] yes
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# ...
```

Listing Filer Shares

It is convenient to show the available back-end-filer shares as you add them into a direct volume. Filer shares are configured from Network-Attached Storage (NAS) filers or file servers with Direct-Attached Storage (DAS). Use the show exports external-filer ... shares command to show a filer's shares using the external-filer name assigned at the ARX. This is very similar to the show exports host command described in Chapter 5, *Examining Filers*:

show exports external-filer *filer* shares
[user *username* windows-domain *domain* | proxy-user *proxy*]

where

filer (1-1024 characters) is the external filer's name, as displayed in show external-filer (see "Listing External Filers" on page 6-6),

username (optional, 1-64 characters) is a Windows username (for CIFS shares),

domain (optional, 1-64 characters) is the above user's Windows domain, and

proxy (1-64 characters) is an alternative to the username/domain credentials. This the name of the proxy-user configuration, as shown by the show proxy-user command (see "Listing All Proxy Users" on page 3-5).

The output shows NFS shares in one table and CIFS shares in another:

- The NFS table shows each export and the NIS netgroup(s) (shown as IP addresses and wildcards) that can access the share. For an export to be eligible for inclusion in the volume, all proxy-IP addresses must be on this list.

- The CIFS table shows two disk-space measures and the serial number for the storage volume behind the share. If two shares have the same serial number, they are assumed to be shares for the same storage volume on the filer.

For example, the following command shows all of the NFS shares on the “nas1” external filer:

```
bstnA6k(gbl)# show exports external-filer nas1 shares
Export probe of filer “nas1” at 192.168.25.21

CIFS Credentials: [anonymous]

Shares:
  NFS
    Path (Owner)                        Access (Status)
    ----------------------------------  ---------------------------
    /vol/vol0                             (Mounted,rsize=32768,wsize=32768)
...
bstnA6k(gbl)# ...
```

Showing Supported Protocols at the Filer

The filer’s supported protocols must include all of the namespace protocols. To check them, use the capabilities keyword in the show exports command:

show exports external-filer *filer* capabilities

This examination does not require a CIFS login, so no Windows credentials are needed.

For example, this command shows that the “das1” filer supports all versions of NFS as well as CIFS:

```
bstnA6k(gbl)# show exports external-filer das1 capabilities
Export probe of filer “das1” at 192.168.25.19

CIFS Credentials: [anonymous]

Capabilities:
  NFS
    Port Mapper    TCP/111, UDP/111
    Mount Daemon   V1 TCP/642, V1 UDP/639, V2 TCP/642, V2 UDP/639, V3 TCP/642, V3 UDP/639
    Server         V2 TCP/2049, V2 UDP/2049, V3 TCP/2049, V3 UDP/2049

  CIFS
    Security Mode  User level, Challenge/response, Signatures disabled
    Server         TCP/445
    Max Request    16644 bytes
bstnA6k(gbl)# ...
```

Identifying the Filer and Share

The next step in configuring a direct share is identifying its source share on an external filer. From gbl-ns-vol-shr mode, use the filer command to use a filer's exported directory as a share:

filer *name* {{nfs | cifs} *share-name*}+ [access-list *list-name*]

where

name (1-64 characters) is the name of the external filer (see "Listing External Filers" on page 6-6),

nfs | cifs is a required choice,

share-name (1-1024 characters) is the NFS export or CIFS share on the filer, and

list-name (optional, 1-64 characters) is the name of an NFS access list to associate with the share (see "Listing All NFS Access Lists" on page 4-10).

Never connect to more than one instance of a given back-end share. If the filer uses aliases for its shares, the ARX assumes that each alias is a different share.

For example, this command set identifies a filer share for the direct share, "corporate." The share is /vol/vol0 on a filer named "nas1:"

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# share corporate
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# show exports external-filer nas1 shares
Export probe of filer "nas1" at 192.168.25.21

CIFS Credentials: [anonymous]
```

```
Shares:
  NFS
    Path (Owner)                          Access (Status)
    ----------------------------------  ---------------------------
    /vol/vol0                             (Mounted,rsize=32768,wsize=32768)
...
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# filer nfs nas1 /vol/vol0
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# ...
```

Identifying a Multi-Protocol Share

In a multi-protocol (NFS and CIFS) namespace, you list both names for the share. You can do this in any order:

filer *name* nfs *nfs-name* cifs *cifs-name* [access-list *list-name*]

or

filer *name* cifs *cifs-name* nfs *nfs-name* [access-list *list-name*]

Disconnecting From the Filer

From gbl-ns-vol-shr mode, use the no filer command to disconnect the current direct share from its filer:

no filer

For example:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# share test
bstnA6k(gbl-ns-vol-shr[medco~/vol~test])# no filer
```

Using a Managed Volume as a Filer

You can assign a managed volume to the share as though it were an external filer. (The next chapter describes how to configure a managed volume.)

Note

If the direct volume's namespace supports CIFS, you can only use a managed volume from the same namespace.

From gbl-ns-vol-shr mode, use the managed-volume command to use a managed volume as a direct share's "filer:"

managed-volume *namespace volume* [access-list *list-name*]

where:

namespace (1-30 characters) is the name of the namespace.

volume (1-1024 characters) is the volume's path (for example, '/rcrds'). You cannot use a direct volume; only a managed volume.

list-name (optional, 1-64 characters) is the NFS access list to associate with the share (see "Listing All NFS Access Lists" on page 4-10).

For example, the following command sequence assigns the 'wwmed~/it' managed volume to the 'test' share.

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# share test
bstnA6k(gbl-ns-vol-shr[medco~/vol~test])# managed-volume wwmed /it
bstnA6k(gbl-ns-vol-shr[medco~/vol~test])# ...
```

Disconnecting from the Managed Volume

Use the no form of the command to stop using a managed volume as a direct share's filer.

no managed-volume

Attaching a Virtual Directory to the Back-End Share

The next step is to create a virtual *attach-point directory*, visible to clients from the root of the volume, and attach it to a physical directory on the back-end filer. For example, you can create an attach point named /vol0 (in the /vol volume) and attach it to the filer's /usr/local directory: the client-viewable /vol/vol0 is then the same as /usr/local on the filer. Clients can mount any level of the volume or attach-point directory (/vol or /vol/vol0). Mounting below that level (for example, /vol/vol0/work) defeats the purpose of federating the attach points, so it is not supported.

From gbl-ns-vol-shr mode, use the attach command to create an attach-point directory and attach it to one of the filer's physical directories:

attach *attach-point-directory* [to *physical-directory*] [access-list *list-name*]

where:

attach-point-directory (1-4096 characters) is the path of the virtual directory that the client sees. This is relative to the root of the volume; for example, if you are in the /home volume and you enter a virtual directory of "aa," clients see this attach point as /home/aa.

to *physical-directory* (optional; 1-1024 characters) specifies the name of the actual directory on the filer share. Similar to the attach-point directory, this path is relative to the root of the filer share (established in the filer command's nfs or cifs clause, above). Use a "." to attach to the root directory in the share. If you omit this clause, the ARX uses the *attach-point-directory* path.

list-name (optional 1-64 characters) is the NFS access list to associate with this attach point (see "Listing All NFS Access Lists" on page 4-10).

You can use the command multiple times in the same direct share, thereby attaching to multiple directories on the back-end share. The maximum number of total attach points is resource- and platform-dependent: a later section describes various limits on resource usage, including attach points.

Attach-point directories cannot be nested: if you create /var in one attach point, you cannot create /var/log in another attach point in the same volume.

For example, this command sequence sets up the filer for the “corporate” share (as shown above), then attaches three directories to the filer:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# share corporate
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# filer nas1 nfs /vol/vol0/direct
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# attach vol0/corp to shr
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# attach vol0/notes to notes
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# attach conCalls
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# ...
```

This creates the following map of virtual to physical directories:

Virtual Directories (seen by clients)	Physical Directories (on filer)
/vol/vol0/corp	/vol/vol0/direct/shr
/vol/vol0/notes	/vol/vol0/direct/notes
/vol/conCalls	/vol/vol0/direct/conCalls

Removing an Attach Point

Use the no attach command to remove one attach point from the share:

no attach *attach-point-directory*

where ***attach-point-directory*** (1-1024 characters) is the path of the virtual directory to detach.

For example, this command sequence detaches the conCalls directory from the “corporate” share:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# share corporate
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# no attach conCalls
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# ...
```

Designating the Share as Critical (optional)

If the current switch has a redundant peer, you have the option to designate the current share as *critical*. Skip to the next section if this switch is not configured for redundancy.

If the direct volume software loses contact with one of its critical (and enabled) shares, the ARX initiates a failover. The failover is accepted by the redundant peer as long as the peer has full access to all critical shares, critical routes, *and* the quorum disk. If the peer is unable to access any of these critical resources, no failover occurs. (For instructions on configuring critical routes, refer back to "Identifying a Critical Route" on page 6-15 of the *CLI Network-Management Guide*.)

From gbl-ns-vol-shr mode, use the critical command to designate the current share as a critical resource:

critical

For example, this command sequence designates the "corporate" share as a critical share:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# share corporate
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# critical
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# ...
```

Removing Critical-Share Status

By default, shares are not critical. If the switch loses contact with a non-critical share, the volume operates in a degraded state and the switch does not initiate a failover. From gbl-ns-vol-shr mode, use the no critical command to make the current share non-critical:

no critical

For example, this command sequence removes the "generic" share from the list of critical shares:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
```

```
bstnA6k(gbl-ns-vol[medco~/vol])# share generic
bstnA6k(gbl-ns-vol-shr[medco~/vol~generic])# no critical
bstnA6k(gbl-ns-vol-shr[medco~/vol~generic])# ...
```

Ignoring the Share’s Free Space (optional)

This option is only relevant in a volume where you are manually calculating free space (recall “Manually Setting the Volume’s Free Space (optional)” on page 8-3). Such a volume’s free space is the sum of the space from *all* of its shares, including multiple shares from the same back-end storage volume. This can mean counting the same storage multiple times: two or more shares from the same storage volume each report the full amount of free space on the volume. For example, two NFS shares from the same disk partition, /lhome, would each report the total free space on the /lhome partition. A volume with both of these shares would double-count the free space in /lhome.

You can manually exclude shares from the free-space calculation. From gbl-ns-vol-share mode, use the freespace ignore command to ignore the free space on the current share:

freespace ignore

You can set this before or after the share and volume are enabled.

For example, this command sequence ignores all free space in the “rec98” share:

```
prtlndA1k(gbl)# namespace access
prtlndA1k(gbl-ns[access])# volume /G
prtlndA1k(gbl-ns-vol[access~/G])# share rec98
prtlndA1k(gbl-ns-vol-shr[access~/G~rec98])# freespace ignore
prtlndA1k(gbl-ns-vol-shr[access~/G~rec98])# ...
```

Including the Share in the Free-Space Calculation

By default, free space from all shares is counted toward the volume’s total. To include this share in the free-space calculation, use no freespace ignore:

no freespace ignore

For example:

```
prtlndA1k(gbl)# namespace access
prtlndA1k(gbl-ns[access])# volume /G
prtlndA1k(gbl-ns-vol[access~/G])# share recsY2k
prtlndA1k(gbl-ns-vol-shr[access~/G~recsY2k])# no freespace ignore
prtlndA1k(gbl-ns-vol-shr[access~/G~recsY2k])# ...
```

Adjusting the Free-Space Calculation

You can also manually adjust the free-space that is advertised for the current share. From gbl-ns-vol-share mode, use the freespace adjust command:

freespace adjust [-]*adjustment*[K|M|G|T]

where:

- (optional) makes the adjustment negative,

adjustment (1-64) is the size of the adjustment, and

K|M|G|T (optional) chooses the unit of measure: **K**ilobytes, **M**egabytes, **G**igabytes, or **T**erabytes. The default is bytes if you omit this. All values are base-2; e.g., a kilobyte is 1,024 bytes and a megabyte is 1,048,576 bytes.

As with freespace ignore, you can set this before or after the share and volume are enabled.

For example, this command sequence increases the free space calculation for the "recs2002" share:

```
prtlndA1k(gbl)# namespace access
prtlndA1k(gbl-ns[access])# volume /G
prtlndA1k(gbl-ns-vol[access~/G])# share recs2002
prtlndA1k(gbl-ns-vol-shr[access~/G~recs2002])# freespace adjust 1G
prtlndA1k(gbl-ns-vol-shr[access~/G~recs2002])# ...
```

Erasing the Free-Space Adjustment

Use the no freespace adjust command to remove any free-space adjustment from the current share:

no freespace adjust

For example, this command sequence removes any free-space adjustment from the "corporate" share:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# share corporate
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# no freespace adjust
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# ...
```

Enabling the Share

The final step in configuring a share is to enable it. An enabled share is an active part of the direct volume; clients can see all the share's attach-point directories after the share is enabled. To enable a share, use the enable command in gbl-ns-vol-shr mode:

enable

In the show namespace output, a direct share has a status of "Online: Direct." In show namespace status, the share's status is abbreviated to "Online."

For example, the following command sequence enables the medco ~/vol~corporate share.

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# share corporate
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# enable
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# ...
```

Disabling the Share

You can disable a direct share to remove its attach points from the volume. This causes all of the share's attach-point directories to effectively disappear from client view. Use no enable in gbl-ns-vol-shr mode to disable the share:

no enable

For example, the following command sequence disables the medco~/vol~sales share.

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
```

```
bstnA6k(gbl-ns-vol[medco~/vol])# share sales
bstnA6k(gbl-ns-vol-shr[medco~/vol~sales])# no enable
bstnA6k(gbl-ns-vol-shr[medco~/vol~sales])# ...
```

Removing a Direct Share

Use the no share command to remove a share from a direct volume:

no share

For example, this command set removes the "test" share from the "/vol" volume in the "medco" namespace:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# no share test
bstnA6k(gbl-ns-vol[medco~/vol])# ...
```

Selecting a VPU (optional)

The next step in configuring a volume is to choose its Virtual-Processing Unit (VPU). A *VPU* is a virtual CPU that can fail over from one chassis to another in a redundant configuration. The namespace software chooses a default VPU if you do not explicitly choose one.

Caution

The default-VPU assignment rules (described next) can artificially reduce the maximum number of namespaces for the host switch. Once the volume is enabled, its VPU assignment is permanent. Read this section carefully before using the default-VPU assignment.

Each VPU runs on its own physical processor during normal operation; after a redundancy failover, a single physical processor can run two VPUs. In the ARX®500 or ARX®1000, a single VPU runs on the ACM and can fail over to its redundant peer's ACM. In an ARX®6000, two VPUs run on each ASM (an ASM has two processors, each of which supports one VPU).

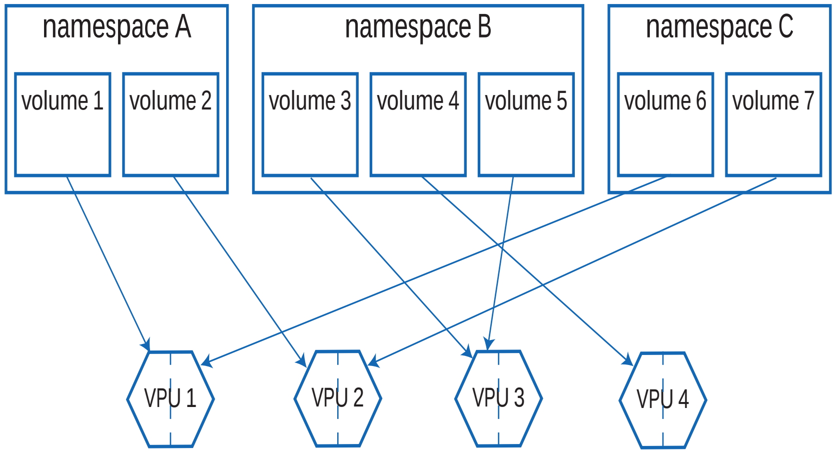
Each VPU supports up to 64 volumes from up to 2 namespaces. You can scale the number of namespaces and volumes on an ARX®6000 by adding more ASMs to the switch.

Table 8-1. Numbers of Supported Volumes per Platform

Platform	# VPUs	# Namespaces	# Volumes
ARX®500	1	2	64
ARX®1000	1	2	64
ARX®6000 with 1 ASM	2	4	128
ARX®6000 with 2 ASMs	4	8	256

Default-VPU Assignment

By default, a newly-enabled volume is assigned to the least-subscribed VPU. If possible, volumes from every namespace are split between two different VPUs. For example, the following diagram illustrates the default assignments for a three namespaces on an ARX®6000 with two ASMs. This assumes that the volumes are enabled from left to right:



The namespace software uses the following rules for assigning a volume to a VPU:

First volume in the namespace

a. Choose an empty VPU.

b. Choose a VPU that is supporting only one namespace.

c. Fail if all VPUs have two namespaces already.

Second volume in the namespace

a. Choose the least-subscribed VPU that is *different* from the first.

b. Choose the same VPU as the first volume.

c. Fail if all VPUs have 64 volumes already.

Third and subsequent volumes in the namespace

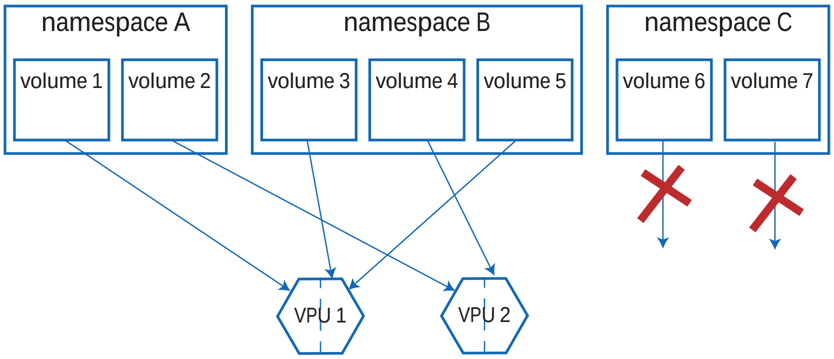
a. Choose a VPU that supports volume 1 or 2; choose the one with the least total volumes.

b. Fail if all VPUs have 64 volumes already.

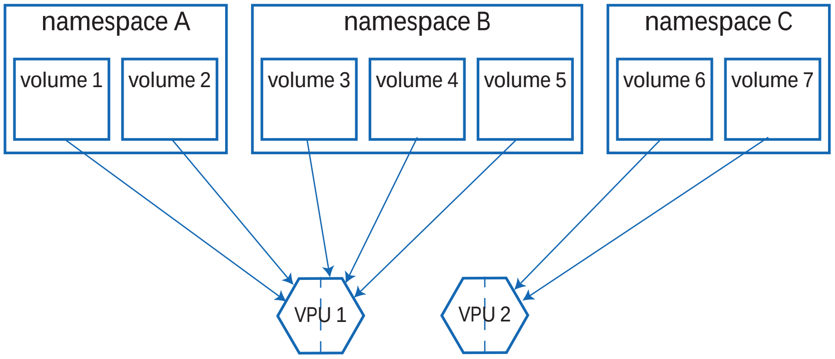
A failure prevents you from enabling the volume.

Assigning the Volume to a VPU

The default-VPU assignment algorithm can artificially limit the maximum number of namespaces on your ARX. Consider the above example with a single ASM. According to Table 8-1, the single ASM has two VPUs and can therefore support up to four namespaces. However, if the namespace volumes are enabled from left to right (as above), the first two namespaces claim both VPUs before any of namespace C's volumes get enabled. The volumes in namespace C can therefore never be enabled.



You can avoid this problem by manually assigning all of the volumes in namespaces A and B to VPU 1, thereby leaving both halves of VPU 2 free for namespace C.



From gbl-ns-vol mode, use the vpu command to assign the volume to a VPU:

vpu *id*

where ***id*** (1-6) identifies the VPU.

Do this *before* the volume is enabled; once the volume is enabled (below), you cannot re-assign it to another VPU.

For example, the following command sequence assigns the current volume, “medco~/vol,” to VPU 1:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# vpu 1
bstnA6k(gbl-ns-vol[medco~/vol])# ...
```

Splitting Namespace Processing within a VPU

Each VPU has two *domains*, one per namespace. By default, volumes are assigned to domains so that the domains run *different* namespaces. You can manually assign the current volume to a particular domain, so that both domains have volumes from the same namespace. (This option is more useful for managed volumes than direct volumes; its use for managed volumes is discussed in the next chapter.)

To assign the current volume to a specific domain, use the optional domain clause at the end of the vpu command:

vpu *id* domain *domain-id*

where

id (1-6) identifies the VPU, and

domain-id (1-2) chooses the VPU domain.

For example, the following command sequence explicitly assigns the current volume, “medco~/vol,” to VPU 1, domain 2:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# vpu 1 domain 2
bstnA6k(gbl-ns-vol[medco~/vol])# ...
```

Reverting to Default-VPU Assignment

Before you enable the direct volume, you can remove the manual-VPU assignment. This causes the namespace software to assign the volume according to the default rules (refer back to "Default-VPU Assignment" on page 8-20). From gbl-ns-vol mode, use the no vpu command to revert to default-VPU assignment:

no vpu

For example:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /test_results
bstnA6k(gbl-ns-vol[medarcv~/test_results])# no vpu
bstnA6k(gbl-ns-vol[medarcv~/test_results])# ...
```

VPU Limits for Direct Volumes and Shares

The number of VPUs on your platform determines the maximum volumes and shares you can add. Table 8-2 specifies limits on volumes and direct-volume shares for each platform.

Table 8-2. VPU Configuration Limits

Platform	Maximum Volumes (all types)	Maximum Shares per Direct Volume	Maximum Direct Shares (total)	Maximum Attach Points
ARX®500 (1 VPU)	64	255	4,096	524,288
ARX®1000 (1 VPU)	64	255	4,096	524,288
ARX®6000 (per VPU)	64	255	4,096	524,288
• ARX®6000 with 1 ASM (2 VPUs)	128	255	8,192	1,048,576
• ARX®6000 with 2 ASMs (4 VPUs)	256	255	16,384	2,097,152

These limits are evaluated on a credit system; to create a new direct volume or share, its VPU must have sufficient credits. Volume limits are enforced whenever a volume is enabled, and share limits are enforced when both the share and its volume are enabled. The enable operation is denied if the VPU lacks sufficient credits.

Showing All VPUs on the Current Switch

Use the show vpu command to see all the current volume-to-VPU assignments:

show vpu [detailed]

where **detailed** (optional) adds some details to the output: CPU and memory usage on the physical processor.

Note

This shows the VPUs on the current switch; if this volume is hosted by the switch's redundant peer, log on to the CLI on that switch to show its VPU assignments.

For each VPU, the output displays the physical processor where the VPU is running, the state of the VPU, the numbers of share/volume/file credits used and remaining, and a table of namespaces and volumes running on the VPU.

For example, the following command shows all VPUs on an ARX®6000.

```
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# show vpu

Switch: bstnA6k
----------------------------------------------------------------------
VPU 1
-----
Physical Processor: 3.1
State: Normal; maximum instances
Share credits: 4 shares used (252 credits remain of total 256)
Direct share credits: 3 direct shares used (4093 credits remain of total 4096)
Volume credits: 2 volumes used (62 credits remain of total 64)
File credits: 4.0M files reserved (380M credits remain of total 384M)

Namespace           Domain   Volume                    State
```

```
---------         ------  ------               -----
medco              2      /vol                 Enabled
wwmed              1      /acct                Enabled

2 Namespaces            2 Volumes

VPU 2
-----
Physical Processor: 3.2
State: Normal; maximum instances
Share credits: 7 shares used (249 credits remain of total 256)
Direct share credits: 5 direct shares used (4091 credits remain of total 4096)
Volume credits: 6 volumes used (58 credits remain of total 64)
File credits: 132M files reserved (252M credits remain of total 384M)

Namespace         Domain  Volume               State
---------         ------  ------               -----
medarcv            1      /rcrds               Enabled
medarcv            1      /lab_equipment       Enabled
medarcv            1      /test_results        Enabled
insur              2      /claims              Enabled

2 Namespaces            4 Volumes

bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

Showing One VPU

To focus on one VPU, use the VPU number with the show vpu command:

show vpu *id* [detailed]

where

id (1-6) identifies the VPU, and

detailed (optional) is explained above.

For example, the following command shows VPU 1, with CPU and memory details:

```
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# show vpu 1 detailed

Switch: bstnA6k
----------------------------------------------------------------------
VPU 1
-----
Physical Processor: 3.1 (1% CPU, 8% MEM)
State: Normal; maximum instances
Share credits: 4 shares used (252 credits remain of total 256)
Direct share credits: 3 direct shares used (4093 credits remain of total 4096)
Volume credits: 2 volumes used (62 credits remain of total 64)
File credits: 4.0M files reserved (380M credits remain of total 384M)

Namespace         Domain  Volume                State
---------         ------  ------                -----
medco              2      /vol                  Enabled
wwmed              1      /acct                 Enabled

2 Namespaces            2 Volumes

bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

Enabling the Volume

The final step in configuring a direct volume is to enable it. From gbl-ns-vol mode, use the enable command to enable the current volume:

enable

Direct volumes have no metadata; they only attach to directories on back-end filers. The enable process invokes no imports, so it is much faster than an enable in a managed volume (described in the next chapter). Use show namespace [status] to monitor the progress of the enable: the volume is enabled when all of its shares have a status of “Online: Direct” (or simply “Online” in the show namespace status output).

For example, this command sequence enables the “/vol” volume in the “medco” namespace:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# enable
bstnA6k(gbl-ns-vol[medco~/vol])# ...
```

Enabling All Shares in the Volume

From gbl-ns-vol mode, you can enable all of the volume’s shares with a single command. Use the enable shares command to do this:

enable shares

For example, the following command sequence enables all shares in the “medco~/vol” volume:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# enable shares
bstnA6k(gbl-ns-vol[medco~/vol])# ...
```

Disabling All Shares

Use no enable shares command to disable each of the volume’s individual shares:

no enable shares

Caution

This is equivalent to disabling the volume, described below. This causes the volume to stop responding to clients; different client applications react to this in different ways. Some may hang, others may log errors that are invisible to the end user.

For example, this command sequence disables all of the shares in the “access~/G” volume:

```
prtlndA1k(gbl)# namespace access
prtlndA1k(gbl-ns[access])# volume /G
```

```
prtlndA1k(gbl-ns-vol[access~/G])# no enable shares
prtlndA1k(gbl-ns-vol[access~/G])# ...
```

Disabling the Volume

You can disable a volume to stop clients from accessing it. Use no enable in gbl-ns-vol mode to disable the volume:

no enable

Caution

This affects client service. As mentioned above, a disabled volume does not respond to clients; different client applications react to this in different ways. Some may hang, others may log errors that are invisible to the end user.

For example, the following command sequence disables the "medco~/vol" volume:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# no enable
bstnA6k(gbl-ns-vol[medco~/vol])# ...
```

Showing the Volume

To show only one volume in a namespace, add the volume clause to show namespace command (described in the namespace chapter):

show namespace *name* volume *volume*

where:

name (1-30 characters) is the name of the namespace, and

volume (1-1024 characters) is the path name of the volume.

For a namespace with multiple volumes, the output shows only the volume chosen. The output otherwise matches that of the show namespace command.

For example, the following command shows the configuration of the ‘medco~/vol’ volume:

```
bstnA6k# **show namespace medco volume /vol**

Namespace “medco” Configuration
Description
Metadata Cache Size: 512 MB

Domain Information
------------------



Supported Protocols
-------------------
  nfsv3-tcp

Participating Switches
----------------------
  bstnA6k (vpu 1) [Current Switch]



Volumes
-------
  /vol

             Volume freespace: 463GB (automatic)
                Metadata size: 28k
                        State: Enabled

                  Host Switch: bstnA6k
                     Instance: 1
                          VPU: 1 (domain 2)
                        Files: 1 used, 31M free

    Share corporate
    Filer                           nas1 [192.168.25.21]
    NFS Export                      /vol/vol0/direct
    Status                          Online
    Critical Share                  Yes
    Free space on storage           45GB (49,157,705,728 B)
    Free files on storage           1M
    Virtual inodes                  16M
    Transitions                     1
    Last Transition                 Wed Apr  4 03:39:50 2007

  Share generic
    Filer                           nas3 [192.168.25.47]
    NFS Export                      /vol/vol2/direct
    Status                          Online
    Free space on storage           383GB (412,072,050,688 B)
    Free files on storage           6M
    Virtual inodes                  256M
    Transitions                     1
    Last Transition                 Wed Apr  4 03:40:08 2007

  Share sales
    Filer                           nas2 [192.168.25.44]
    NFS Export                      /vol/vol1/direct
    Status                          Online
    Free space on storage           34GB (36,967,763,968 B)
    Free files on storage           6M
    Virtual inodes                  32M
    Transitions                     1
    Last Transition                 Wed Apr  4 03:40:06 2007


bstnA6k# ...
```

Showing One Share

To show the configuration and status of one share in a volume, add the share clause after the volume clause:

show namespace *name* volume *volume* share *share-name*

where:

name (1-30 characters) is the name of the namespace,

volume (1-1024 characters) is the path name of the volume, and

share-name (1-64 characters) identifies the share.

This output shows the share that you chose in the command along with its volume and namespace.

For example, the following command shows the configuration of the ‘medco~/vol~corporate’ share:

```
bstnA6k# show namespace medco volume /vol share corporate

Namespace “medco” Configuration
Description
Metadata Cache Size: 512 MB

Domain Information
------------------



Supported Protocols
-------------------
  nfsv3-tcp

Participating Switches
----------------------
  bstnA6k (vpu 1) [Current Switch]
```

```
Volumes
-------
  /vol

             Volume freespace: 463GB (automatic)
                Metadata size: 28k
                        State: Enabled

                  Host Switch: bstnA6k
                     Instance: 1
                          VPU: 1 (domain 2)
                        Files: 1 used, 31M free

    Share corporate
      Filer                          nas1 [192.168.25.21]
      NFS Export                     /vol/vol0/direct
      Status                         Online
      Critical Share                 Yes
      Free space on storage          45GB (49,157,705,728 B)
      Free files on storage          1M
      Virtual inodes                 16M
      Transitions                    1
      Last Transition                Wed Apr  4 03:39:50 2007


bstnA6k# ...
```

Showing Filer Shares Behind One Volume

You can use the show namespace mapping command to show the filer shares behind a particular namespace, as described in the namespace chapter. This shows all attach points in a direct volume and the physical directories behind them. Add the volume clause to show only the shares behind a particular volume:

show namespace mapping *name* volume *volume*

where:

name (1-30 characters) is the name of the namespace, and

volume (1-1024 characters) is the path name of the volume.

The output shows every attach point in the volume, along with the physical directory to which it connects.

For example, this shows the filer shares behind the “medco~/vol” volume:

```
bstnA6k# show namespace mapping medco volume /vol

Namespace                Physical Server
---------------------    ---------------------
medco:/vol
  vol0/corp              nas1:/vol/vol0/direct/shr
  vol0/notes             nas1:/vol/vol0/direct/notes
  vol1/mtgMinutes        nas2:/vol/vol1/direct/mtgs
  vol1/sales             nas2:/vol/vol1/direct/export
  vol2                   nas3:/vol/vol2/direct/data



Where * denotes metadata only physical server.
bstnA6k# ...
```

Showing the Volume's Configuration

To review the configuration settings for a direct volume, identify the volume at the end of the the show global-config namespace command:

show global-config namespace *namespace volume*

where

namespace (1-30 characters) identifies the namespace, and

volume (1-1024 characters) is the volume.

The output shows all of the configuration options required to recreate the volume. The options are in order, so that they can be used as a CLI script.

For example, the following command shows the configuration for the "medco~/vol" volume:

```
bstnA6k# show global-config namespace medco /vol
;=============================== namespace ===============================
namespace medco
  protocol nfs3tcp
  volume /vol
    direct
    hosted-by bstnA6k
    share corporate
      critical
      filer nas1 nfs /vol/vol0/direct
      attach vol0/notes to notes
      attach vol0/corp to shr
      enable
      exit

    share generic
      filer nas3 nfs /vol/vol2/direct
      attach vol2 to data
      enable
      exit
    share sales
      filer nas2 nfs /vol/vol1/direct
      attach vol1/sales to export
      attach vol1/mtgMinutes to mtgs
      enable
      exit

    vpu 1 domain 2
    enable
    exit

  exit

bstnA6k# ...
```

Sample - Configuring a Direct Volume

For example, this command set configures the ‘/vol’ volume on the ‘medco’ namespace:

```
bstnA6k(gbl)# namespace medco
bstnA6k(gbl-ns[medco])# volume /vol
bstnA6k(gbl-ns-vol[medco~/vol])# direct
bstnA6k(gbl-ns-vol[medco~/vol])# share corporate
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# filer nas1 nfs /vol/vol0/direct
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# attach vol0/corp to shr
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# attach vol0/notes to notes
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# critical
bstnA6k(gbl-ns-vol-shr[medco~/vol~corporate])# exit
bstnA6k(gbl-ns-vol[medco~/vol])# share sales
bstnA6k(gbl-ns-vol-shr[medco~/vol~sales])# filer nas2 nfs /vol/vol1/direct
bstnA6k(gbl-ns-vol-shr[medco~/vol~sales])# attach vol1/sales to export
bstnA6k(gbl-ns-vol-shr[medco~/vol~sales])# attach vol1/mtgMinutes to mtgs
bstnA6k(gbl-ns-vol-shr[medco~/vol~sales])# exit
bstnA6k(gbl-ns-vol[medco~/vol])# share generic
```

```
bstnA6k(gbl-ns-vol-shr[medco~/vol~generic])# filer nas3 nfs /vol/vol2/direct
bstnA6k(gbl-ns-vol-shr[medco~/vol~generic])# attach vol2 to data
bstnA6k(gbl-ns-vol-shr[medco~/vol~generic])# exit
bstnA6k(gbl-ns-vol[medco~/vol])# vpu 1 domain 2
bstnA6k(gbl-ns-vol[medco~/vol])# enable
bstnA6k(gbl-ns-vol[medco~/vol])# show namespace status medco

Namespace: medco
Description:

     Share                     Filer                                      Status
         NFS Export
    ------------------------- ------------------------------------- -----------

  Volume: /vol                                                      Enabled
     corporate                 nas1                                       Online
         NFS: /vol/vol0/direct

     sales                     nas2                                       Online
         NFS: /vol/vol1/direct

     generic                   nas3                                       Online
         NFS: /vol/vol2/direct

bstnA6k(gbl-ns-vol[medco~/vol])# exit
bstnA6k(gbl-ns[medco])# ...
```

Removing a Direct Volume

You can remove a direct volume that is not exported by any front-end service; a later chapter describes how to export or share a volume.

From priv-exec mode, you can use the remove namespace ... volume command to remove a volume:

> **remove namespace *name* volume *volume* [timeout *seconds*] [sync]**
>
> where:
>
> ***name*** (1-30 characters) is the name of the namespace,
>
> ***volume*** (1-1024 characters) is the path name of the volume,
>
> ***seconds*** (optional, 300-10,000) sets a time limit on each of the removal's component operations, and
>
> **sync** (optional) waits for the removal to finish before returning. With this option, the CLI lists the volume components as it removes them.

The CLI prompts for confirmation before removing the volume. Enter **yes** to continue.

This operation generates a report, "removeNs_*namespace_date*.rpt," which catalogs all of the actions that it took. The *namespace* in the filename identifies the removed namespace, and the *date* is the date and time when the command started. The CLI shows the report name after you invoke the command. Use show reports to see the file listing; use show, tail, or grep to read the file. To save the report off to an external site, use the copy command from priv-exec mode. The command does not create the report if you use the sync option; it shows its actions at the command line instead.

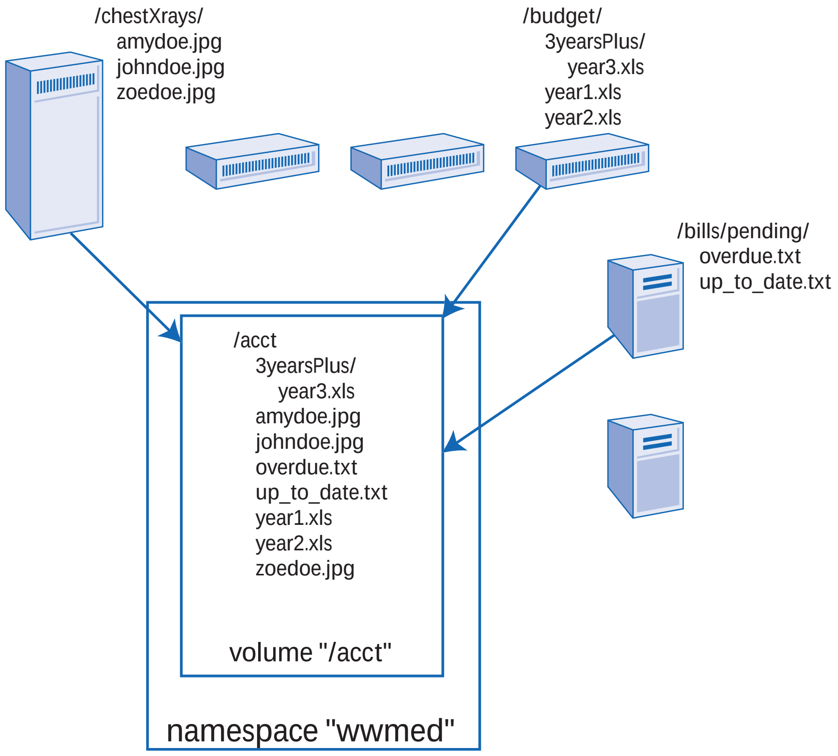
For example, this command sequence removes the '/trialvol' volume from the 'medco' namespace:

```
bstnA6k(gbl)# end
bstnA6k# remove namespace medco volume /trialvol
...
```

Chapter 9

Adding a Managed Volume

A *managed volume* aggregates one or more exports/shares from actual filers. The files from each filer share are *imported* into the top directory of the volume. During the share import, the volume catalogues all file and directory locations in its *metadata*. For example, an "/acct" volume with shares from three filers would aggregate files as shown in the figure below:



Metadata facilitates storage policies, but it requires some management. A direct volume, described in the previous chapter, has no metadata and is therefore easier to set up and tear down.

As explained in the namespace chapter, you use the gbl-ns volume command to create a volume (see "Adding a Volume" on page 7-21). This puts you into gbl-ns-vol mode, where you configure import parameters, metadata storage, at least one share, and several of the options discussed earlier for direct volumes.

For example, this command set creates a single volume ("/acct") for the "wwmed" namespace:

```
bstnA6k(gbl)# namespace wwmed

bstnA6k(gbl-ns[wwmed])# volume /acct

This will create a new volume.


Create volume '/acct'? [yes/no] yes

bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Storing Volume Metadata on a Dedicated Share

A managed volume stores its metadata on a reliable external share. This share, called a *metadata share*, is devoted exclusively to Acopia metadata. Best practices dictate that you use a dedicated metadata share for each managed volume. The namespace can have a single metadata share (as described in the *CLI Reference Guide*) to be divided amongst its managed volumes, but a dedicated share for each volume is preferred.

Caution

It is vitally important that the external metadata share is fast, highly available, and has multiple gigabytes of free space. A managed volume cannot function if it loses contact with its metadata share.

An NFS metadata share works for either an NFS or CIFS volume. A CIFS metadata share is not as flexible; an NFS-only volume cannot use a CIFS metadata share because the volume does not have the required proxy-user credentials (see "Configuring Windows Authentication (CIFS)" on page 7-14) to access a CIFS share.

From gbl-ns-vol mode, use the metadata share command to use a dedicated metadata share for the current volume:

metadata share *filer* {nfs3 | nfs3tcp | cifs} *path*

where

filer (1-64 characters) is the name of the external filer,

nfs3 | nfs3tcp | cifs chooses the protocol to access the share (this can be **nfs3** or **nfs3tcp** for a CIFS-only volume), and

path (1-1024 characters) is the specific export/share on the filer. This external share cannot be used as a namespace share. Use a unique metadata share for each volume. Do not use /vol/vol0 on a NetApp filer; this is reserved for the NetApp operating system, and is not intended for fast access.

For example, this command sequence sets a metadata share to hold all metadata for the “wwmed~/acct” volume:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# metadata share nas1 nfs3 /vol/vol1/meta1
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Using Multiple Metadata Shares

You can create one or more fallback metadata shares in case of a problem with the first share (for example, the share goes offline) during the volume’s import. If the volume software fails to initialize metadata at the first metadata share, it uses the next configured share instead.

For example, the following command sequence configures the “medarcv~/rcrds” volume with three candidates for metadata. During the import, the volume software tries each share in the same order as entered here (first nas1, then nas6, then nas3):

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# metadata share nas1 nfs3 /vol/vol1/meta
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# metadata share nas6 nfs3tcp /vol/vol2/aco_md
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# metadata share nas3 cifs acp_meta
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

Only one share is chosen during the import, and the volume uses that share to store metadata as long as it runs. After the metadata share is chosen, the volume ignores all of the remaining metadata shares.

Removing a Metadata Share

You can remove an unused metadata share from a managed volume.

Caution

A managed volume requires metadata storage for a successful import. Before you do this, verify that the volume has at least one metadata share before it is enabled.

To remove a metadata share, use the no metadata share command from gbl-ns-vol mode:

no metadata share *filer* {nfs3 | nfs3tcp | cifs} *path*

where

filer (1-64 characters) is the name of the filer,

nfs3 | nfs3tcp | cifs is the file-access protocol, and

path (1-1024 characters) is the specific export/share on the filer.

This command has no effect after the volume has already imported its metadata.

For example, this command sequence removes the metadata share on nas3 from the "/rcrds" volume:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# no metadata share nas3 cifs acp_meta
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

Designating the Metadata Share as Critical (optional)

If the current switch has a redundant peer, you have the option to designate the volume's metadata share as *critical*. Skip to the next section if this switch is not configured for redundancy.

If the volume software loses contact with its metadata share, the ARX initiates a failover. The failover is accepted by the redundant peer as long as the peer has full access to all critical shares, critical routes, critical metadata shares, *and* the quorum disk. If the peer is unable to access any of these critical resources, no failover occurs. (For instructions on configuring critical routes, refer to "Identifying a Critical Route" on page 6-15 of the *CLI Network-Management Guide*.)

If the switch has a redundant peer, we recommend that you use this option for all managed volumes. Without metadata, the managed volume cannot function.

From gbl-ns-vol mode, use the metadata critical command to designate the current volume's metadata share as a critical resource:

metadata critical

For example, this command sequence designates the metadata share as a critical resource for the "nemed~/acctShdw" volume:

```
prtlndA1k(gbl)# namespace nemed
prtlndA1k(gbl-ns[nemed])# volume /acctShdw
prtlndA1k(gbl-ns-vol[nemed~/acctShdw])# metadata critical
prtlndA1k(gbl-ns-vol[nemed~/acctShdw])# ...
```

Removing Critical-Resource Status

By default, metadata shares are not critical. If the managed volume loses contact with its metadata share in this case, the volume fails and the switch does not initiate a failover. This is not recommended for redundant switches. For non-redundant switches, it is the only option.

From gbl-ns-vol mode, use the no metadata critical command to make the metadata share non-critical for the current volume:

no metadata critical

For example:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# no metadata critical
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Migrating Metadata to a New Share After Import

After the managed volume is fully enabled, it chooses its metadata share and writes several database files onto it. You may discover that the metadata filer is not as fast or reliable as you would prefer. In this case, you can migrate the volume's metadata to a new filer and share using the Namespace Check (nsck) tool. (The nsck tool is described in the *CLI Maintenance Guide*.)

Dividing the Import into Multiple Scans

By default, the volume imports its metadata by scanning files and directories together. Client access to the volume is limited during import: clients can read directories, add files, and edit files; but they cannot add, delete, rename, or make a hard link to any directories, nor can they rename, hard-link to, or delete files. This is appropriate when you have an adequate cut-in window for the ARX, when client access is extremely limited or completely disallowed for the duration of the import.

You can allow full client access sooner by setting the volume to separately scan for directories first, then for files. This import is slower than the single-scan import, but it allows full client access after the faster directory scan is finished. Additionally, clients are allowed to delete files through both scans. This table summarizes client-access permissions during the various scans.

Scan Type	Read or Traverse	Create or Write	Rename	Delete	Hard-Link to
Single Scan	Files + Directories	Files only	Neither	Neither	Neither
Directory Scan	Files + Directories	Files only	Neither	Files only	Neither
File Scan	Files + Directories				

A multi-scan import is appropriate for an installation with a short cut-in window. From gbl-ns-vol mode, use the import multi-scan command to separate the file scan from the directory scan:

import multi-scan

This does not have any affect on an import that is currently underway; it applies to future imports only.

For example, this command sequence prepares the “medarcv~/lab_equipment” volume for a multi-scan import:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /lab_equipment
bstnA6k(gbl-ns-vol[medarcv~/lab_equipment])# import multi-scan
bstnA6k(gbl-ns-vol[medarcv~/lab_equipment])# ...
```

Protecting Metadata During Import

A fully-imported volume protects its metadata by recording all metadata transactions in battery-backed, Non-Volatile RAM (NVRAM). After an unexpected reboot or failover, the volume can retrieve these transaction records from NVRAM and use them to recover all in-flight metadata changes. By default, the volume speeds up its import by skipping these metadata-protection measures until the import is complete. An unexpected reboot or failover during import therefore causes the import to restart. NFS clients who may have been connected during the import would have to re-mount the volume because the re-import changes all NFS filehandles. CIFS clients would not have a noticeable interruption in service.

You can set the volume to protect its metadata during its next import. This guards against the unlikely event of a reboot, but at the cost of a slower import. From gbl-ns-vol mode, use the import protection command to protect volume metadata during import:

import protection

As with the import multi-scan command, this does not affect a currently-running import.

For example, this command sequence protects the metadata in the “wwmed~/acct” volume during import:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# import protection
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Reverting to Unprotected Metadata and Faster Import

Protected metadata introduces a performance penalty during import. An unprotected import is often the best choice when it fits comfortably into the assigned cut-in window. As mentioned above, CIFS clients would be affected far less than NFS clients in the event of an unexpected reboot. To leave metadata unprotected during import, thereby making import faster, use the no import protection command:

no import protection

After the import is over, the volume starts recording all metadata transactions in NVRAM. The metadata is only unprotected during the import.

For example, this command sequence lifts all metadata protection from the “medarcv~/lab_equipment” volume during import:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /lab_equipment
bstnA6k(gbl-ns-vol[medarcv~/lab_equipment])# no import protection
bstnA6k(gbl-ns-vol[medarcv~/lab_equipment])# ...
```

Reverting Back to the Faster, Single-Scan Import

If the cut-in window is adequate for the single-scan import time, you can use it to shorten the overall time for import. Use no import multi-scan to revert to single-scan imports:

no import multi-scan

As mentioned above, this only applies to future imports.

For example, this command sequence reverts the “wwmed~/acct” volume to the faster single-scan import:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
```

```
bstnA6k(gbl-ns-vol[wwmed~/acct])# no import multi-scan
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Safe Modes for Share Imports into Pre-Enabled Volumes

After the managed volume is fully enabled, a newly added share always uses the multi-scan import with metadata protection. This is regardless of the volume-level settings for import multi-scan and/or import protection, which only apply to a full volume import. A full volume import occurs when it is initially enabled, re-enabled after being destaged with the Namespace Check (nsck) tool, or rebuilt with the nsck tool. (The nsck tool is described in the *CLI Maintenance Guide*.)

Allowing the Volume to Modify on Import

When a managed volume imports its shares, it is possible for the shares to have redundant filenames (for example, the /etc/hosts file could exist on two different NFS shares). These are called file *collisions*, and they can prevent the import from succeeding. The import can only succeed if you allow the volume to rename the second file and import it with the new name. If such modifications are allowed, the second file is renamed according to the following convention:

```
filename_share-jobid.ext
```

where

filename is the file's original name, without its extension (for example, "myfile" from "myfile.doc"),

share is the name of the namespace share (described below),

jobid is an integer identifying the job that imported the share, and

.ext is the file's original extension (for example, ".doc"), if there is one.

If there is more than one redundant file, an index is added:

```
filename_share-jobid-index.ext
```

where *index* is a number starting at 1.

Redundant directories are only a problem if their file attributes (such as their permissions settings) do not match, or if they have the same name as an already-imported file. Collided directories are renamed according to the same convention as files.

Note

For a multi-protocol (NFS and CIFS) namespace, directories are also renamed if their CIFS names are not mappable to the NFS-character encoding. This was discussed in "Setting NFS Character Encoding" on page 7-11.

You will have the option to disallow file renames and/or directory renames for individual shares. These are discussed later in the share-configuration section.

From gbl-ns-vol mode, use the modify command to allow the volume to modify redundant files and/or directories on import:

modify

Clients cannot write to the volume until you run the modify command.

The CLI prompts with a question about re-imports. The *CLI Maintenance Guide* describes how to use a namespace-check tool (nsck) to re-import a previously-enabled volume. This tool turns off the modify setting and takes the volume offline. The modify setting stays off during re-import unless you raise a flag by answering this question. Answer "yes" to allow the modify setting to remain if the volume is rebuilt through nsck.

For example, this command sequence allows the '/acct' volume to modify files on import and re-import.

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# modify
Automatically re-enable volume modify mode during nsck rebuild? [yes/no] yes
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Conditions for Running the modify Command

You can run the modify command

- before the volume is first enabled, or

- after an import with no modify (assuming no file or directory collisions occurred on import).

You cannot use the modify command if the volume is in the process of importing, if any imported shares have collisions, or if the nsck utility is being used on the volume.

Running a No-Modify Import

If you choose to import without using the modify command (to run a hypothetical import) and there are no file collisions, you can later raise the modify flag. This completes the import and gives write access to clients.

You can use the shares' import reports to find any collisions that would have occurred. Import reports appear after the managed volume is enabled. Use show reports to find all reports on the system; import reports are named "import.*share-id*.*share-name*.*job-id*.rpt." Use show reports *name* to view the report.

Allowing the Volume to Modify on Re-Import

As mentioned above, the *CLI Maintenance Guide* describes the nsck tool for rebuilding a managed volume. If you said "no" to the CLI prompt after you used the modify command, nsck does not raise the modify flag after taking the volume offline. This turns modifications off after a volume rebuild. You can use another CLI command, reimport-modify, to allow the modify flag to stay up after the volume is rebuilt:

reimport-modify

This command prompts for confirmation before taking action; enter **yes** to proceed.

For example, this command sequence ensures that the "/acct" volume will be modifiable after any re import:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# reimport-modify

Automatically re-enable volume modify mode during NSCK rebuild? [yes/no] yes
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Preventing Modification On or After Re-Import

Use the no reimport-modify command to keep the modify flag down after using nsck:

no reimport-modify

This is the default.

For example:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# no reimport-modify
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Preventing Modifications

By default, the managed volume is read-only after its initial import, and it does not modify any back-end files during the import. Use no modify command to return to this default:

no modify

You can only run this command on a volume with no imported shares. That is, you can run this before you first enable the volume, or after taking the volume offline with the nsck tool.

For example:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# no modify
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Automatically Synchronizing Metadata (CIFS)

This section only applies to volumes in namespaces that support CIFS. Skip to the next section if the namespace is NFS-only.

If a file changes on a filer without the managed volume's knowledge, the volume's metadata is compromised. This can happen if one of the filer's local applications, such as anti-virus software, deletes a file or moves it to a quarantine area. If a client tries to access the file through the volume, the client gets an error. You can configure the managed volume to automatically detect this error and synchronize its metadata with the actual files. From gbl-ns-vol mode, use the auto sync files command to allow the volume to automatically synchronize its metadata:

auto sync files

For example, the following command sequence allows the "medarcv~/rcrds" volume to automatically sync its metadata:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# auto sync files
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

Auto-Sync Jobs and Reports

The volume launches a sync job whenever a filer returns an error indicating that the volume's metadata is wrong. Each auto-sync job gets a unique job ID, similar to the job IDs for import jobs. It also generates a report; use the show reports command to list all reports or to examine an auto-sync report's contents. All auto-sync reports are named as follows:

`auto-sync.`*`job-id`*`.`*`vol-path`*`.rpt`

where

job-id is an integer identifying the job that synchronized the volume, and

vol-path is the volume's name, where all slashes (/) are replaced with underscores (_).

Allowing Renames on Collision

An auto-sync job may discover a file that collides with another file in the volume (that is, in another share). By default, this prevents the operation from synchronizing that file; a managed volume cannot support two or more files whose path names collide. To work around these collisions, you can configure the volume to rename these files before using them. The volume uses the following pattern to rename the file:

```
filename_share-jobid.ext
```

where

filename is the file’s original name, without its extension (for example, “zapFiles” from “zapFiles.exe”),

share is the name of the volume share (described below),

jobid is an integer identifying the job that synchronized the volume, and

.ext is the file’s original extension (for example, “.exe”), if there is one.

If there is more than one redundant file, an index is added:

```
filename_share-jobid-index.ext
```

where *index* is a number starting at 1.

This renaming convention is the same one that is used for file collisions on import (recall “Allowing the Volume to Modify on Import” on page 9-9).

From gbl-ns-vol mode, use the rename-files flag at the end of the auto sync files command to allow the volume to rename any collision files:

auto sync files rename-files

For example, the following command sequence allows sync-file renames in the “medarcv~/rcrds” volume:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# auto sync files rename-files
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

Disallowing Renames

If auto-sync jobs are not allowed to rename files that collide, those files cannot be synchronized. The metadata for those files remains stale, so clients cannot access them. To disallow renames, use the no auto sync files rename-files command:

no auto sync files rename-files

For example,

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# no auto sync files rename-files
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

Disabling File Auto-Synchronization

The *CLI Maintenance Guide* describes a manual *sync* utility that you can use to synchronize metadata on command. You can use this utility as needed. From gbl-ns-vol mode, use no auto sync files to stop the automatic syncs and rely exclusively on the manual utility:

no auto sync files

For example:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /itvol
bstnA6k(gbl-ns-vol[medarcv~/itvol])# no auto sync files
bstnA6k(gbl-ns-vol[medarcv~/itvol])# ...
```

Manually Setting the Volume's Free Space (optional)

The next step in creating a volume is to choose an algorithm for calculating its free space. This is the free-space calculation that is passed onto the client: whenever a user mounts a volume (NFS) or maps a network drive to it (CIFS), this total is the free space that they see. By default, the volume automatically detects any back-end shares that draw from the same storage pool, and counts the free space in only one of those back-end shares.

The direct-volume chapter described how to manually configure free space in a direct volume. Recall "Manually Setting the Volume's Free Space (optional)" on page 8-3. The command is the same for a managed volume: freespace calculation manual from gbl-ns-vol mode. For example, this makes the 'nemed~/acctShdw' volume count the free space in all back-end shares, even multiple shares that draw from the same back-end storage pool:

```
prtlndA1k(gbl)# namespace nemed
prtlndA1k(gbl-ns[nemed])# volume /acctShdw
prtlndA1k(gbl-ns-vol[nemed~/acctShdw])# freespace calculation manual
prtlndA1k(gbl-ns-vol[nemed~/acctShdw])# ...
```

You can then adjust the free space at each share, as described in the direct-volume chapter and re-iterated later in this chapter.

Using Automatic Free-Space Calculation

By default, free space is calculated based on the IDs of the volumes behind the back-end shares. If any of these shares report the same volume ID, their free space is counted only once. To return to this default, use no freespace calculation manual. For example:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# no freespace calculation manual
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Setting CIFS Options

The next step in configuring a volume is addressing its CIFS options, if necessary. Skip to the next section if this volume is in an NFS-only namespaces.

There are five CIFS-volume attributes that back-end filers may or may not support. They are named streams, compressed files, persistent ACLs, Unicode file names on disk, and sparse files. Each volume can support any and all of these capabilities. However, *all* filer shares used in a volume *must* support the capabilities advertised in the volume. For example: if your volume is going to support compressed files, then *all* of its back-end filer shares must also support compressed files. The show exports command displays the CIFS options on back-end filers, as described in "Showing CIFS Attributes" on page 5-13.

The CIFS options conform to those of the first-enabled share by default; the options are scanned during the share import. To manually control these options, you can use the same commands that you use in a direct volume (recall "Setting CIFS Options" on page 8-4). That is, you can use any combination of the following gbl-ns-vol commands to manually set the options:

[no] named-streams

[no] compressed-files

[no] persistent-acls

[no] unicode-on-disk

[no] sparse-files

For example, the following command sequence disables two CIFS options in the "medarcv~/test" volume:

```
bstnA6k(gbl)# namespace medarcv

bstnA6k(gbl-ns[medarcv])# volume /test

bstnA6k(gbl-ns-vol[medarcv~/test])# no named-streams

bstnA6k(gbl-ns-vol[medarcv~/test])# no sparse-files

bstnA6k(gbl-ns-vol[medarcv~/test])# ...
```

Supporting Filers with Local Groups

A Windows filer can support *Global Groups*, which are managed by Domain Controllers, and/or *Local Groups*, which are unique to the filer. Local groups have their own Security IDs (SIDs), unknown to any other Windows machine. When you aggregate shares from these filers into a single volume, some files tagged for local-group X are likely to migrate to another filer, which does not recognize the SID for that group (SID X). This invalidates any Access Control Entries (ACEs) with SID X while the file resides on that filer; members of group X lose their privileges. To resolve this problem, you must first prepare all of the filers before you aggregate them into a namespace volume:

- all local-group names must be configured on all filers behind the volume, and
- all groups must contain the same users on those filers.

For example, if filer A has local group “doctors” and filer B has local group “nurses,” you must add a new “nurses” local group to filer A and a new “doctors” group to filer B. The membership of these groups must match at both filers. This preparation is required so that all doctors and nurses can share the filers behind a volume.

Disabling CIFS Oplocks (optional)

The CIFS protocol supports *opportunistic locks* (oplocks) for its files. A client application has the option to take an oplock as it opens a file. While it holds the oplock, it can write to the file (or a cached copy of the file) knowing that no other CIFS client can write to the same file. Once another client tries to access the file for writes, the server gives the first client the opportunity to finish writing. This feature makes it possible for clients to cache their file-writes locally, thereby improving client performance.

CIFS volumes provide oplock support by default. Some installations prefer not to offer oplocks to their CIFS clients. A volume with oplocks disabled does not grant any oplocks to any CIFS clients, so CIFS clients cannot safely use file caching as described above. As with direct volumes (recall “Disabling CIFS Oplocks (optional)” on page 8-6), you use the cifs oplocks-disable command to disable oplock support in a managed volume. For example, the following command sequence disables oplock support in “insur~/claims,” a multi-protocol (CIFS and NFS) namespace:

```
bstnA6k(gbl)# namespace insur

bstnA6k(gbl-ns[insur])# volume /claims
```

```
bstnA6k(gbl-ns-vol[insur~/claims])# cifs oplocks-disable
bstnA6k(gbl-ns-vol[insur~/claims])# ...
```

Allowing the Volume to Automatically Disable Oplocks

You can configure the managed volume to automatically disable oplocks for a CIFS client that times out in response to an “oplock break” command. The “oplock break” command informs a client that it must finish its writes and release the oplock, so that another client can write to the file. For each client that does not respond in 10 seconds or less, the volume disables oplocks for 10 minutes, then re-enables oplocks for that client and tries again. The volume manages oplocks on a client-by-client basis.

To permit the volume to disable oplocks for misbehaving clients, use the optional auto flag in the cifs oplocks-disable command. For example, the following command sequence allows the “medarcv~/rcrds” volume to automatically disable oplocks:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# cifs oplocks-disable auto
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

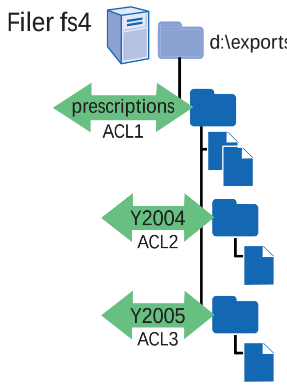
Reinstating Oplock Support

Use no cifs oplocks-disable to support CIFS oplocks in the current managed volume. For example, the following command sequence enables oplocks in the “/rcrds” volume:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# no cifs oplocks-disable
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

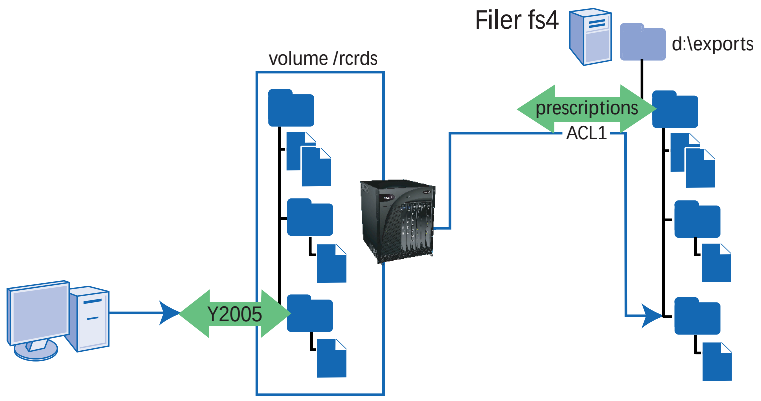
Supporting Subshares and their ACLs

Windows filers can share multiple directories in the same tree, and can apply a different share-level Access Control List (ACL) to each of them. Consider the following three shares on the fs4 filer:



A client who connects to the “prescriptions” share has the access rights defined in ACL1, whereas the same client uses ACL3 if he or she connects to the “Y2005” *subshare*. ACL1 may restrict the client to read-only access, while ACL3 gives the gives the client full control.

By default, all managed-volume access comes through the top-level share at the back-end filer (“prescriptions” in this example). The volume’s CIFS clients would therefore always have the permissions defined by ACL1, even if they connected through the front-end equivalent of the filer’s Y2005 subshare:



To prepare the managed volume to pass connections through to the filer's subshares, thereby using the subshares' ACLs, use the gbl-ns-vol filer-subshares command:

filer-subshares

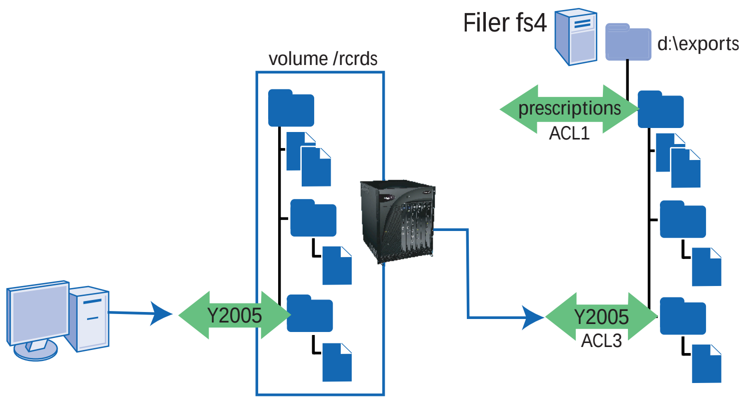
You cannot use this command while any of the volume's shares are enabled.

This command only prepares the volume for subshare support at the back-end. A later chapter describes how to configure the client-visible subshares in a front-end CIFS service.

For example, the following command sequence prepares the "/rcrds" volume to support filer subshares and their back-end ACLs:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# filer-subshares
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

This causes the managed volume to recognize subshares. When you later configure a "Y2005" subshare in the front-end CIFS service, clients who connect to the subshare will use the correct share-level ACLs:



Required Windows Permissions

To read the share and ACL definitions at the back-end filers, the volume requires proxy-user credentials with admin-level access. This is a significant increase in access from the standard proxy-user requirements; you may need to increase the permissions for the proxy user on all filers, or use a more-powerful proxy user for this namespace. For instructions on editing a proxy user, recall “Adding a Proxy User” on page 3-2. “Configuring Windows Authentication (CIFS)” on page 7-14 describes how to add new proxy-user credentials to the namespace.

Showing a Filer’s Shares and Subshares

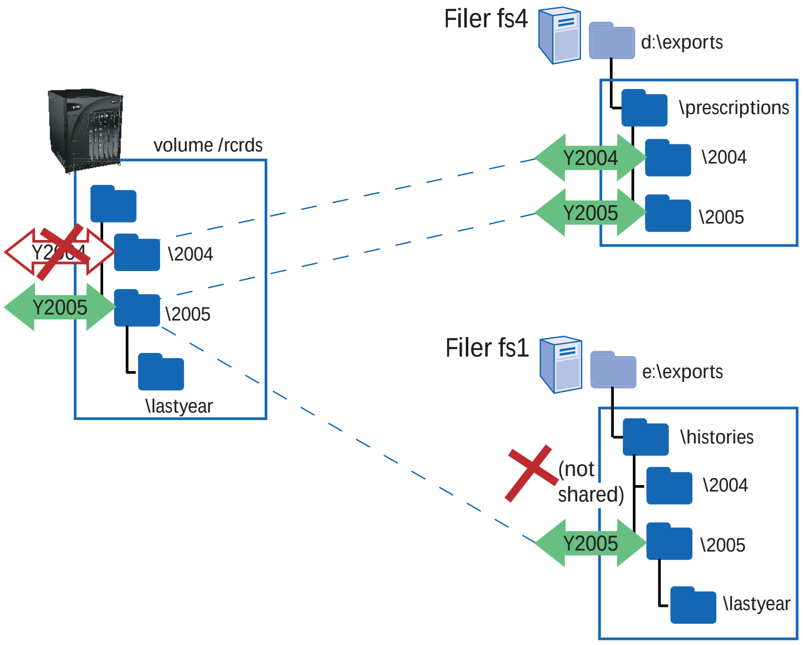
You can use the show exports ... path command to display the physical paths for all CIFS shares on a back-end filer. This was discussed in an earlier chapter: recall “Showing the Physical Paths for CIFS Shares” on page 5-8. The output from this command shows all of the share-to-subshare relationships on the filer. For example, this command shows one share (“prescriptions”) and its three subshares (“CELEBS$,” “Y2004,” and “Y2005”) on the “fs4” filer:

```
bstnA6k(gbl)# show exports external-filer fs4 paths proxy-user acoProxy2
Export probe of filer “fs4” at 192.168.25.29

CIFS Credentials: MEDARCH\jqpublic

Paths:

  CIFS

    Share                          Directory
    ---------------------------    -------------------------------------
    CELEBS$                        d:\exports\prescriptions\VIP_wing
    Y2004                          d:\exports\prescriptions\2004
    Y2005                          d:\exports\prescriptions\2005
    prescriptions                  d:\exports\prescriptions
bstnA6k(gbl)# ...
```

Replicating Subshares at all of the Volume's Filers

The managed volume must have consistent subshares and subshare ACLs under all of its back-end shares. Consistency is required so that clients have the same access point and permissions no matter which back-end share contains their files and directories. If a subshare definition is missing from any share, or has a different ACL, the volume cannot import the top-level share.

For example, suppose the \2004 directory is shared on fs4 only while the \2005 directory is shared on both fs4 and fs1. The /rcrds volume can only support the subshare that is shared on both filers. This prevents the volume from importing the top-level share on fs1:



You can arrange for the volume to correct this during share import, so that it copies all subshare definitions from the first-imported share to the remaining shares. If necessary, the volume copies underlying directories as well. To set this import option, use the optional replicate flag in the filer-subshares command:

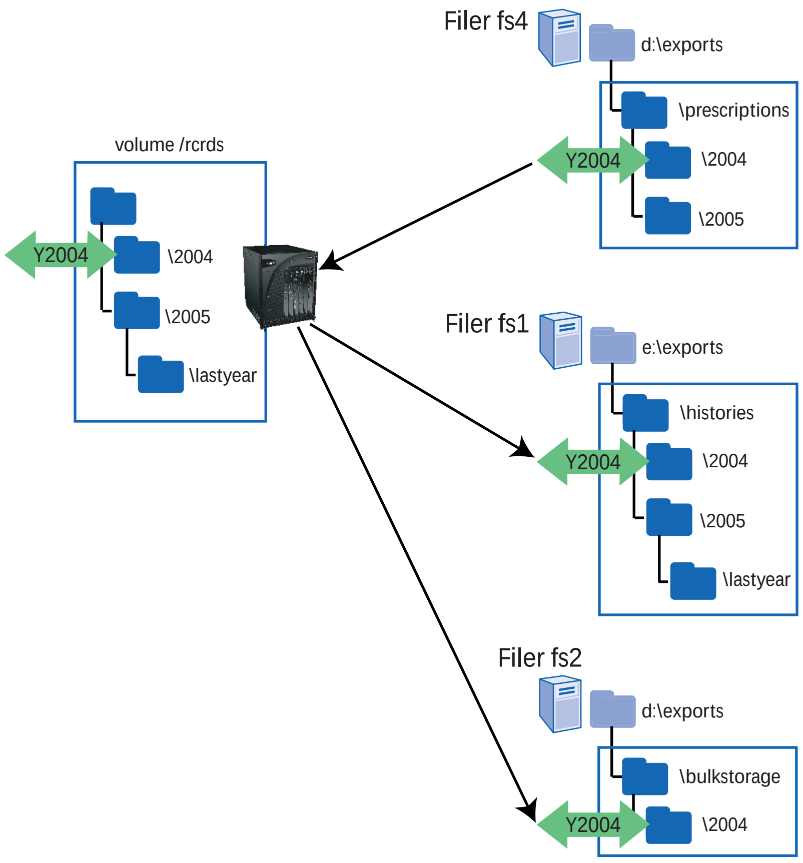
```
filer-subshares replicate
```

You can issue this command in an enabled volume that already supports filer subshares. In this case, the volume replicates all subshares to any newly-added shares.

For example, this command sequence replicates all subshares to new shares in the “/rcrds” volume:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# filer-subshares replicate
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

When each share is enabled (as described later), the volume reads all subshare information from the first share and replicates it to the remaining shares. The “Y2004” subshare is therefore supported:



Disabling Filer Subshares

From gbl-ns-vol mode, use the no filer-subshares command to disable volume support for CIFS subshares and their share-level ACLs:

no filer-subshares

You can only disable this feature when no CIFS services are sharing any of the volume's subshares. CIFS front-end services are described in a later chapter, along with instructions on sharing CIFS subshares.

For example, this command sequence prevents CIFS-subshare support in the "insur~/claims" volume:

```
bstnA6k(gbl)# namespace insur
bstnA6k(gbl-ns[insur])# volume /claims
bstnA6k(gbl-ns-vol[insur~/claims])# no filer-subshares
bstnA6k(gbl-ns-vol[insur~/claims])# ...
```

Adding a Share

The next step in creating a managed volume is identifying one or more shares for it. A *share* is one CIFS share or NFS export from an external (NAS or DAS) filer. A volume can contain multiple shares, where each typically resides on a different filer.

Note

The first configured share in a managed volume, by default, will hold all new files created in the volume's root. The volume's *root* is the volume's top-most directory (for example, /acct). The share is called the *root-backing* share for the volume. Choose the root-backing share carefully, as it could possibly be overburdened by new files from clients.

To ease this burden, you can configure a *share farm* to distribute new files among multiple shares. You will learn how to configure these policies in a later chapter.

As with direct volumes, you use the share command to add a share to a managed volume. This puts you into gbl-ns-vol-shr mode, where you must identify the filer and export/share, and then you must enable the share. A managed-volume share also has several options related to import.

For example, this command set adds a share called “bills” to the “/acct” volume in the “wwmed” namespace:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share bills
This will create a new share.

Create share 'bills'? [yes/no] yes
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# ...
```

Showing Available Filer Shares

You need to know the names and protocols of your configured filer shares to import one into your managed volume. Use the show exports external-filer ... shares command to show a filer’s shares (recall “Listing Filer Shares” on page 8-8), or use the show exports host ... shares command described in Chapter 5, *Examining Filers*. The first syntax identifies the filer by its external-filer name, the second uses the filer’s IP address.

For example, the following command shows that there is only one share on the “das8” external filer:

```
bstnA6k(gbl)# show exports external-filer das8 shares
Export probe of filer “das8” at 192.168.25.25
% INFO: Filer 192.168.25.25 does not support CIFS or is unreachable.

CIFS Credentials: [anonymous]

Shares:
  NFS
    Path (Owner)                         Access (Status)
    ----------------------------------  ---------------------------
    /work1                               * (Mounted,rsize=32768,wsize=32768)
bstnA6k(gbl)# ...
```

Identifying the Filer and Share

The most important step in configuring a share is connecting it to an export/share on an external filer. The export/share must support all of the namespace's protocols; a CIFS namespace can only import CIFS shares, and an NFSv3 namespace can only import NFSv3 exports.

You use the filer command to identify the filer share behind the managed share. This is the same as the filer command for a direct-volume share (recall "Identifying the Filer and Share" on page 8-10).

For example, this command set identifies an export for the "bills" share. The export is /work1/accting, on the "das8" filer:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share bills
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# filer das8 nfs /work1/accting
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# ...
```

Identifying a Multi-Protocol Share

For a multi-protocol namespace, you list both names for the share. You can do this in any order:

filer *name* nfs *nfs-name* cifs *cifs-name* [access-list *list-name*]

or

filer *name* cifs *cifs-name* nfs *nfs-name* [access-list *list-name*]

For example, the following command sequence uses a multi-protocol filer for the "insur~/claims~shr1-old" share:

```
bstnA6k(gbl)# namespace insur
bstnA6k(gbl-ns[insur])# volume /claims
bstnA6k(gbl-ns-vol[insur~/claims])# share shr1-old
bstnA6k(gbl-ns-vol-shr[insur~/claims~shr1-old])# filer nas1 cifs insurance nfs
/vol/vol1/NTFS-QTREE/insurance
bstnA6k(gbl-ns-vol-shr[insur~/claims~shr1-old])# ...
```

Disconnecting From the Filer Before the Share is Enabled

To correct a mistake, you can disconnect a share from its filer before you enable the share. (The process of enabling a share is described later.) Use the no filer command described earlier in "Disconnecting From the Filer" on page 8-11: For example, this command set disconnects the filer from the "spreadsheets" share:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share spreadsheets
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~spreadsheets])# no filer
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~spreadsheets])# ...
```

Disconnecting From the Filer After the Share is Enabled

After you enable the share, it has files and directories that were *imported* from the filer. That is, files and directories were scanned and incorporated into the volume's metadata. The *CLI Maintenance Guide* describes how to remove an already-imported share without disrupting any client access to the share's files or directories.

Setting Share-Import Options

Each managed-volume share has several options that you can set to control its import. These subsections describe each of them in detail.

Speeding Up Import by Skipping a Directory Test

A managed volume tests each imported directory to verify that it is not already imported into another managed volume. Two volumes cannot manage the same directory. This managed-volume check is crucial for shares that may have been previously imported by any ARX, so it is enabled by default. For a new filer, the first introduction of the ARX at the site, or an nsck *rebuild* of the volume (described in the *CLI Maintenance Guide*), you can disable this check to speed up the import of the share. From gbl-ns-vol-shr mode, use import skip-managed-check to skip the directory check:

import skip-managed-check

For example, the following command sequence allows the "medarcv~/lab_equipment" volume to skip this check while importing the 'equip' share.

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /lab_equipment
bstnA6k(gbl-ns-vol[medarcv~/lab_equipment])# share equip
bstnA6k(gbl-ns-vol-shr[medarcv~/lab_equipment~equip])# import skip-managed-check
bstnA6k(gbl-ns-vol-shr[medarcv~/lab_equipment~equip])# ...
```

Reinstating the Directory Test

If there is any doubt about any directory in the share, the volume should verify that none of them are managed by some other volume. Use the no import skip-managed-check command to re-instate the directory check:

no import skip-managed-check

For example, the following command sequence sets the /acct volume to check all the directories in the 'bills' share.

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share bills
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# no import skip-managed-check
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# ...
```

Synchronizing Directory Attributes on Import

Two directories *collide* when they have the same name but different file attributes, such as permission settings. If the managed volume has modify enabled (see "Allowing the Volume to Modify on Import" on page 9-9), it renames any directory that collides with an already-imported directory. For an individual share, you can choose an alternative: instead of renaming the directory, synchronize its attributes with that of its already-imported counterpart. The volume presents the two directories as a single directory, with the aggregated contents of both and the attributes of the one that was imported first.

Note: For heterogeneous multi-protocol namespaces, always enable synchronization with the import sync-attributes command. A multi-protocol volume compares both CIFS and NFS attributes, thereby dramatically increasing the likelihood of directory collisions and renames.

For a directory that collides with an already-imported *file*, a rename is the only possible method for resolving the conflict.

From gbl-ns-vol-shr mode, use the import sync-attributes command to allow directory-attribute synchronization for the current share.

import sync-attributes

For example, the following command sequence allows the /acct volume to synchronize directory attributes (instead of renaming conflicting directories) in the ‘bills’ share.

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share bills
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# import sync-attributes
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# ...
```

Disabling Directory-Attribute Synchronization on Import

By default, if two shares have matching directories but conflicting attributes, the directories *collide*. Like a file collision, this is handled by renaming the directory as specified by the modify flag. Use the no import sync-attributes command to disable the synchronization:

no import sync-attributes

For example, the following command sequence turns off attribute synchronization in the ‘bills’ share during an import:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
```

```
bstnA6k(gbl-ns-vol[wwmed~/acct])# share bills
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# no import sync-attributes
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# ...
```

Preventing Directory Renames During Import

Whether or not the managed volume is allowed to synchronize attributes on this share, it has occasion to rename any directories that collide with previously-imported *files*. Attribute synchronization is not enough to resolve a naming conflict between a directory and a file. The renamed directory follows the same convention described earlier; recall "Allowing the Volume to Modify on Import" on page 9-9.

Each share generates a report while it imports, and this report includes the original name and the new name for each renamed directory.

You can use the no import rename-directories command to protect this share against any directory renames.

no import rename-directories

If you set this, a directory/file conflict causes the share to fail its import, and a directory/directory conflict fails the import *unless* the volume is allowed to synchronize the attributes of the directories (as shown above). The *CLI Maintenance Guide* explains how to remove a share from a namespace if its import fails; after that, you can directly connect to the back-end share and repair the directories that collided (change names, reset attributes, and so forth).

For example, the following command sequence prevents the volume from renaming any colliding directories in the 'bills' share during an import:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share bills
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# no import rename-directories
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# ...
```

Allowing Directory Renames on Import

If the share allows directory renames, the volume renames its colliding directories as specified by the modify command (refer back to "Allowing the Volume to Modify on Import" on page 9-9). From gbl-ns-vol-shr mode, use the import rename-directories command to permit the volume to rename this share's directories:

import rename-directories

This is the default setting.

For example, the following command sequence returns the 'bills' share to its default:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share bills
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# import rename-directories
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# ...
```

Renaming Directories with Non-Mappable Characters (Multi-Protocol)

This section applies to volumes in a multi-protocol (CIFS and NFS) namespace only. Skip to the next section if the volume supports only CIFS or only NFS.

If directory renames are allowed (as described above), you can set an additional option for renaming directories with non-mappable characters. (Non-mappable characters are discussed in the namespace chapter; see "Setting NFS Character Encoding" on page 7-11.) By default, any directory with a non-mappable character in its name causes the share import to fail. This default prevents accidental renames. You can ensure a successful import by allowing the volume to rename the directories. The volume uses same basic renaming syntax as with any other directory, but replaces each non-mappable CIFS character with its numeric Unicode equivalent:

```
new-dirname_share-jobid[-index][.ext]
```

where *new-dirname* contains "(U+*nnnn*)" in place of each non-mappable character. The *nnnn* is the Unicode number for the character, shown in hexadecimal format. The name is truncated if it exceeds 256 characters. For example, "dir(U+30D2)(U+30AA)_myshare-2."

The resulting name is visible through NFS and CIFS, and can be correlated to the intended CIFS name for the directory. As mentioned above, you can look at the share's import report to see the original name and the new name for each renamed directory.

Use the unmapped-unicode option with the import rename-directories command to allow the volume to rename directories with non-mappable characters:

import rename-directories unmapped-unicode

For example, the following command sequence allows the "insur~/claims" volume to rename any directories with non-mappable characters:

```
bstnA6k(gbl)# namespace insur
bstnA6k(gbl-ns[insur])# volume /claims
bstnA6k(gbl-ns-vol[insur~/claims])# share shr1-old
bstnA6k(gbl-ns-vol-shr[insur~/claims~shr1-old])# import rename-directories
unmapped-unicode
bstnA6k(gbl-ns-vol-shr[insur~/claims~shr1-old])# ...
```

Preventing File Renames During Import

If the volume is allowed to modify files and directories on import, it renames files that collide with previously-imported files or directories. You can prevent the volume from renaming files in this share, causing the share's import to instead fail on file collisions. From gbl-ns-vol-shr mode, use the no import rename-files command to prevent any file renames in the current share:

no import rename-files

The *CLI Maintenance Guide* explains how to remove a share from a namespace if its import fails (see "Removing an Imported Share" on page 8-14 of that manual). Once the share is removed from the namespace, you can directly connect to the back-end share and repair the files that collided (change names, reset attributes, and so forth).

For example, the following command sequence disallows file renames in the 'bills' share during an import.

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share bills
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# no import rename-files
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# ...
```

Allowing File Renames in Import

If the share allows file renames, the volume renames its colliding files as specified by the modify command.

import rename-files

This is the default setting.

For example, the following command sequence returns the ‘bills’ share to its default:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share bills
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# import rename-files
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# ...
```

Enabling SID Translation for a Share (CIFS)

This section only applies to a share that supports CIFS; skip to the next section if this share is in an NFS-only namespace.

A Windows filer can support *Global Groups*, which are managed by Domain Controllers, and/or *Local Groups*, which are unique to the filer. Local groups have their own Security IDs (SIDs), unknown to any other Windows machine. When you aggregate shares from these filers into a single volume, some files tagged for local-group X are likely to migrate to another filer, which does not recognize the SID for that group (SID X). This invalidates any Access Control Entries (ACEs) with SID X while the file resides on that filer; members of group X lose their privileges. To resolve this problem, you must first prepare all of the filers before you aggregate them into a namespace volume. This was discussed earlier; recall “Supporting Filers with Local Groups” on page 9-18.

Once the filers are prepared, you can use the CLI to tag their shares for SID translation. This causes the volume to translate SIDs for all files that migrate to or from these shares: it finds the group name at the source share (such as “nurses” on filer B), then looks up the SID for that group name (“nurses”) at the destination share (on filer A).

Each share whose filer uses Local Groups must have SID translation enabled. For each of these shares, enter gbl-ns-vol-shr mode and use the sid-translation command:

sid-translation

For example, the following command sequence configures one share in the /rcrds volume for SID translation:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# share bulk
bstnA6k(gbl-ns-vol-shr[medarcv~/rcrds~bulk])# sid-translation
bstnA6k(gbl-ns-vol-shr[medarcv~/rcrds~bulk])# ...
```

Disabling SID Translation

Use no sid-translation to stop the volume from translating SIDs for the share. This implies that the share is backed by a filer that uses global SIDs from the DC.

no sid-translation

For example, the following command sequence disables SID translation for the “rx” share:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# share rx
bstnA6k(gbl-ns-vol-shr[medarcv~/rcrds~rx])# no sid-translation
bstnA6k(gbl-ns-vol-shr[medarcv~/rcrds~rx])# ...
```

Finding SID Translations at All Filers

From gbl-ns-vol mode, use the show sid-translation command to show the mapping between SIDs and names on all of the volume’s shares.

show sid-translation *principal*

where ***principal*** (1-256 characters) is a group name, user name, or SID to translate.

The output displays the translation at each share. For example, the following command sequence discovers that the shares behind the 'medarcv~/rcrds' volume have different SIDs for the 'pharmacists' group:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# show sid-translation pharmacists
SID Translations:

  Share rx
    SID:  S-1-5-21-1454471165-492894223-682003330-1006
    Name: MEDARCH\pharmacists (group)

  Share charts
    SID:  S-1-5-21-1454471165-492894223-682003330-1006
    Name: MEDARCH\pharmacists (group)

  Share bulk
    SID:  S-1-5-21-2006398761-1897008245-3502739112-1007
    Name: PV770N\pharmacists (alias)

bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

Ignoring SID Errors from the Filer (CIFS)

If the share does not perform SID translation, or if SID translation fails, the share copies the binary version of the SID to the destination filer. It is possible that the SID is unknown at the destination filer. Typically, the back-end filer returns an error and rejects the file, or silently accepts the unknown SID: an error indicates that the file was rejected. The share therefore aborts the migration if it receives any of the following errors in response to the CIFS 'set security descriptor' command: STATUS_INVALID_SID, STATUS_INVALID_OWNER, or STATUS_INVALID_PRIMARY_GROUP.

Some file servers issue these errors for unknown SIDs but accept the file anyway. Some EMC file servers have this setting as a default. As long as the file server is configured to accept the file or directory (perhaps erasing the unknown SIDs), the volume can safely ignore these errors.

Caution

Do *not* ignore any SID errors from a file server that rejects the file or directory. The SID errors alert the volume to the failed migration. If the volume ignores SID errors from such a file server, file and/or directory loss may result.

To ignore SID errors from the file server, use the ignore-sid-errors command from gbl-ns-vol-shr mode:

ignore-sid-errors

For example, the following command sequence ignores all SID errors from the file server behind the "insur~/claims~shr1-next" share:

```
bstnA6k(gbl)# namespace insur
bstnA6k(gbl-ns[insur])# volume /claims
bstnA6k(gbl-ns-vol[insur~/claims])# share shr1-next
bstnA6k(gbl-ns-vol-shr[insur~/claims~shr1-next])# ignore-sid-errors
bstnA6k(gbl-ns-vol-shr[insur~/claims~shr1-next])# ...
```

Acknowledging SID Errors

It is unsafe to ignore SID errors from a filer or file server that is properly configured. If SID errors truly indicate that the file was rejected, the ARX share should acknowledge them and cancel the file migration. To acknowledge SID errors from the filer, use the no ignore-sid-errors command:

no ignore-sid-errors

For example, the following command sequence acknowledges all SID errors from the filer behind the "insur~/claims~shr1-old" share:

```
bstnA6k(gbl)# namespace insur
bstnA6k(gbl-ns[insur])# volume /claims
bstnA6k(gbl-ns-vol[insur~/claims])# share shr1-old
bstnA6k(gbl-ns-vol-shr[insur~/claims~shr1-old])# no ignore-sid-errors
bstnA6k(gbl-ns-vol-shr[insur~/claims~shr1-old])# ...
```

Designating the Share as Critical (optional)

If the current switch has a redundant peer, you have the option to designate the current share as *critical*. Skip to the next section if this switch is not configured for redundancy.

If the volume software loses contact with one of its critical (and enabled) shares, the ARX initiates a failover. The failover is accepted by the redundant peer as long as the peer has full access to all critical shares, critical routes, *and* the quorum disk. If the peer is unable to access any of these critical resources, no failover occurs.

As explained in the direct-volume chapter (recall "Designating the Share as Critical (optional)" on page 8-15), you can use the critical command to designate the current share as a critical resource. For example, this command sequence designates the "bills" share as a critical share:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share bills
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# critical
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# ...
```

Removing Critical-Share Status

By default, shares are not critical. If the switch loses contact with a non-critical share, the volume operates in a degraded state and the switch does not initiate a failover. Use the no critical command described earlier ("Removing Critical-Share Status" on page 8-15). For example, this command sequence removes the "bills2" share from the list of critical shares:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share bills2
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills2])# no critical
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills2])# ...
```

Ignoring the Share’s Free Space (optional)

This option is only relevant in a volume where you are manually calculating free space (see “Manually Setting the Volume’s Free Space (optional)” on page 9-16). Such a volume’s free space is the sum of the space from *all* of its shares, including multiple shares from the same back-end storage volume. This can mean counting the same storage multiple times: two or more shares from the same storage volume each report the full amount of free space on the volume. For example, two NFS shares from the same disk partition, /lhome, would each report the total free space on the /lhome partition. A volume with both of these shares would double-count the free space in /lhome.

You can manually exclude shares from the free-space calculation using freespace ignore, as described for direct shares (recall “Ignoring the Share’s Free Space (optional)” on page 8-16). For example, this command sequence ignores all free space in the “back2” share:

```
prtlndA1k(gbl)# namespace nemed
prtlndA1k(gbl-ns[nemed])# volume /acctShdw
prtlndA1k(gbl-ns-vol[nemed~/acctShdw])# share back2
prtlndA1k(gbl-ns-vol-shr[nemed~/acctShdw~back2])# freespace ignore
prtlndA1k(gbl-ns-vol-shr[nemed~/acctShdw~back2])# ...
```

Including the Share in the Free-Space Calculation

By default, free space from all shares is counted toward the volume’s total. To include this share in the free-space calculation, use no freespace ignore as you would with a direct share (recall “Including the Share in the Free-Space Calculation” on page 8-16). For example:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share bills
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# no freespace ignore
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# ...
```

Adjusting the Free-Space Calculation

You can also manually adjust the free-space that is advertised for the current share. From gbl-ns-vol-share mode, use the freespace adjust command. This was described in detail for direct volumes; see “Adjusting the Free-Space Calculation” on page 8-17. For example, this command sequence adds 1 gigabyte to the “back1” share’s free space calculation:

```
prtlndA1k(gbl)# namespace nemed
prtlndA1k(gbl-ns[nemed])# volume /acctShdw
prtlndA1k(gbl-ns-vol[nemed~/acctShdw])# share back1
prtlndA1k(gbl-ns-vol-shr[nemed~/acctShdw~back1])# freespace adjust 1G
prtlndA1k(gbl-ns-vol-shr[nemed~/acctShdw~back2])# ...
```

Erasing the Free-Space Adjustment

Use the no freespace adjust command to remove any free-space adjustment from the current share. For example, this command sequence removes any free-space adjustment from the “back2” share:

```
prtlndA1k(gbl)# namespace nemed
prtlndA1k(gbl-ns[nemed])# volume /acctShdw
prtlndA1k(gbl-ns-vol[nemed~/acctShdw])# share back2
prtlndA1k(gbl-ns-vol-shr[nemed~/acctShdw~back2])# no freespace adjust
prtlndA1k(gbl-ns-vol-shr[nemed~/acctShdw~back2])# ...
```

Enabling the Share

The final step in configuring a share is to enable it. An enabled share is an active part of the managed volume; the volume uses enabled shares in its storage pool, and excludes all disabled shares. As with a direct-volume share, use the enable command in gbl-ns-vol-shr mode.

Note

If this is a CIFS volume and it replicates subshares (recall “Replicating Subshares at all of the Volume’s Filers” on page 9-23), all subshares in the first-enabled share are copied to any shares that you enable later. If you want any subshares to be used in the volume, access the back-end filer directly and add them before you enable the first share.

For example, the following command sequence enables the “wwmed ~/acct~bills” share.

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share bills
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# enable
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# ...
```

If the managed volume is also enabled (as described below), it begins to import files and directories from the back-end share. You can use the show namespace status command to monitor the progress of the import. You can look ahead to “Monitoring the Import” on page 9-55 for details.

Taking Ownership of the Share (optional)

Before a managed volume imports the share, it checks the root directory in the back-end share for a special file that marks it as “owned” by an ARX. If this marker file exists, the managed volume does not proceed with the import; no two volumes can manage the same share. You may need to override this safety mechanism for a special case.

Consider an installation that uses legacy, filer-based applications to prepare for disaster recovery: it copies all of its directories and files from a primary site to the filers at another site. If an ARX manages the directories at the primary site, it places its ownership marker in the root of each share. The filer-based application copies the marker files to the remote site, along with all data files. An ARX at the backup site cannot import these shares because of the markers.

You can use the optional take-ownership flag for this special case. If the managed volume finds an ownership marker in the root of this back-end share, it overwrites the marker file. Otherwise, it imports the share as usual:

enable take-ownership

Caution

Do not use this option if it is possible that another ARX is managing this back-end share. This would unexpectedly remove the share from service at the other ARX.

The CLI prompts for confirmation before taking ownership of the share. Enter **yes** to proceed.

For example, the following command sequence enables the "insur_bkup~/insurShdw~backInsur" share, taking ownership of the share if necessary:

```
prtlndA1k(gbl)# namespace insur_bkup
prtlndA1k(gbl-ns[insur_bkup])# volume /insurShdw
prtlndA1k(gbl-ns-vol[insur_bkup~/insurShdw])# share backInsur
prtlndA1k(gbl-ns-vol-shr[insur_bkup~/insurShdw~backInsur])# enable take-ownership
This command allows the switch to virtualize shares that are used by other Acopia switches.
Allow switch to take ownership of share? [yes/no] yes
prtlndA1k(gbl-ns-vol-shr[insur_bkup~/insurShdw~backInsur])# ...
```

Examining the shareEnableSubshareInc Report (CIFS)

This only applies to CIFS volumes where filer-subshares is enabled; recall "Supporting Subshares and their ACLs" on page 9-20. Skip this section unless this option is active in the volume.

Whenever you enable a share in such a CIFS volume, the volume generates a report on all subshare-related activity. The CLI shows the name of the report after you issue the enable command. For example:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# share charts
bstnA6k(gbl-ns-vol-shr[medarcv~/rcrds~charts])# enable

% INFO: Successfully replicated all consistent back-end subshares to 'charts'. See report
'shareEnableSubshareInc_200612201446.rpt' for details.

bstnA6k(gbl-ns-vol-shr[medarcv~/rcrds~charts])# ...
```

This report shows the progress of any replications or subshare-consistency checks that the volume performed. Use the show reports *report-name* command to view the report. For example, this shows the report from the above share-enable command:

```
bstnA6k(gbl-ns-vol-shr[medarcv~/rcrds~charts])# show reports
shareEnableSubshareInc_200612201446.rpt
**** Share Enable Filer-Subshare Inconsistency Report: Started at Wed Dec 20 14:46:50 2006
****
**** Software Version: 2.05.000.09900 (Dec 19 2006 17:39:42) [nbuilds]
**** Hardware Platform: ARX-6000
```

```
The following changes were made to replicate nested shares and their
attributes to the new share:

  Added share “CELEBS$” to the following filer:

    Filer Name: fs1
    IP Address: 192.168.25.20
    Path:       d:\exports\histories\VIP_wing


  Added share “Y2004” to the following filer:

    Filer Name: fs1
    IP Address: 192.168.25.20
    Path:       d:\exports\histories\2004


  Added share “Y2005” to the following filer:

    Filer Name: fs1
    IP Address: 192.168.25.20
    Path:       d:\exports\histories\2005

**** Total processed:               3
**** Elapsed time:           00:00:03
**** Share Enable Filer-Subshare Inconsistency Report: DONE at Wed Dec 20 14:46:53 2006
****
bstnA6k(gbl-ns-vol-shr[medarcv~/rcrds~charts])# ...
```

This sample report shows that the volume successfully replicated three subshares. Later, all three subshares can be exported from a front-end CIFS service.

For a volume that supports filer subshares but does not replicate them, the report would have shown which subshares and ACLs were discovered on the filer. All subshares must match in terms of their directory paths (relative to the share root), share names, and ACLs. If a subshare definition is inconsistent at the share, this report describes the inconsistency. The inconsistent subshare is unusable by the managed volume. You can access the filer directly to make the subshare/ACL consistent. Then you can use the priv-exec sync shares command to add the subshare(s) into the managed volume. For details on the sync shares command, see “Adding and Synchronizing Filer Subshares (CIFS)” on page 5-30 of the *CLI Maintenance Guide*.

Disabling the Share

You can disable a share to make it inaccessible to namespace clients. This stops access to all files on the share. As in a direct volume, use no enable in gbl-ns-vol-shr mode to disable the share.

Caution

This suspends all policy rules in the current volume; the rules are enabled, but not enforced. To bring policy back online for the current volume, remove the share (described below) or re-enable it.

Removing a Managed-Volume Share

You can easily remove a share before its volume is first enabled and starts importing files. Use no share in this case, just as describe for a share in a direct volume. For example, this command set removes the “billsBU” share from the “/acct” volume in the “wwmed” namespace:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# no share billsBU
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Removing the Share After it is Enabled

After you enable the share, it has files and directories that were imported from the filer. That is, files and directories were scanned and incorporated into the volume’s metadata. These files are visible to the volume’s clients, and cannot be removed without disrupting client service. A client-friendly approach is to migrate the files to other shares in the same volume; clients can access the files throughout the migration. The *CLI Maintenance Guide* explains how to migrate all the files and remove an imported share with a single command (see “Removing an Imported Share” on page 8-14 of that manual).

Selecting a VPU (optional)

The next step in configuring a volume is to choose its Virtual-Processing Unit (VPU). A *VPU* is a virtual CPU that can fail over from one chassis to another in a redundant configuration. Every volume type, including managed and direct, runs on a particular VPU. The namespace software chooses a default VPU if you do not explicitly choose one.

Caution

The default-VPU assignment rules can artificially reduce the maximum number of namespaces for the host switch. Once the volume is enabled, its VPU assignment is permanent. This is a much bigger problem for managed volumes than it is for direct volumes. Read about VPUs carefully before using the default-VPU assignment.

VPUs, maximum volumes per platform, maximum namespaces per platform, and the algorithm for default-VPU assignment were discussed in the chapter about direct volumes; recall “Selecting a VPU (optional)” on page 8-19.

Assigning the Volume to a VPU

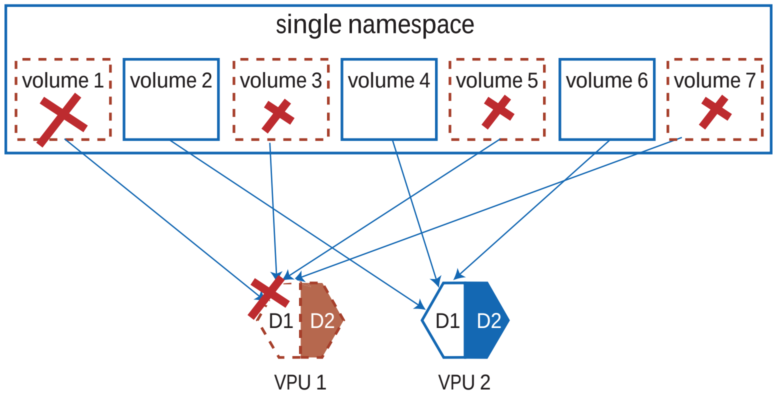
As you would in a direct volume, use the vpu command to assign the volume to a VPU. Do this *before* the volume is enabled; once the volume is enabled (below), you cannot re-assign it to another VPU without entirely rebuilding it.

For example, the following command sequence assigns the current volume, “medarcv~/rcrds,” to VPU 2:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# vpu 2
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

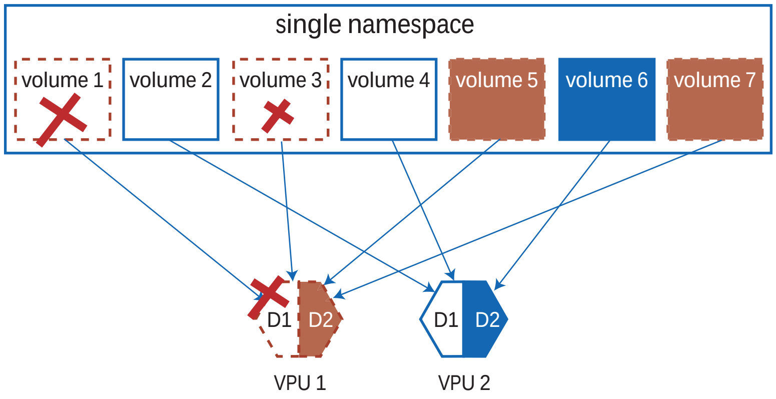
Splitting Namespace Processing within a VPU

Each VPU has two *domains*, one per namespace. If the metadata share fails badly for one volume in a VPU domain, the other volumes in the same domain also fail. For example, consider a system with 7 volumes in a single namespace, divided between 2 VPU domains. A severe metadata failure for volume 1 also affects volumes 3, 5, and 7:



This occurs naturally with default-VPU assignment, but it is undesirable in installations with a large number of volumes in one or two namespaces.

To mitigate this problem, you can assign the same namespace to both VPU domains. This divides the namespace's volumes between the domains. Each domain runs independently; one can have a metadata failure without affecting the other. In the example below, a metadata failure for volume 1 now only affects volume 3:



The volumes on domain 2, volume 5 and volume 7, are completely insulated from the problem.

By default, a volume goes to the same VPU domain as any other volumes from the same namespace. As with a direct volume, you can use the optional domain clause in the vpu command to choose a domain for the current managed volume. For example, the following command sequence assigns the current volume, "medarcv~/test_results," to VPU 2, domain 2:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /test_results
bstnA6k(gbl-ns-vol[medarcv~/test_results])# vpu 2 domain 2
bstnA6k(gbl-ns-vol[medarcv~/test_results])# ...
```

Reverting to Default-VPU Assignment

Before you enable the volume, you can remove the manual-VPU assignment. This causes the namespace software to assign the volume according to the default rules (refer back to "Default-VPU Assignment" on page 8-20). From gbl-ns-vol mode, use the no vpu command to revert to default-VPU assignment:

no vpu

For example:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /test_results
bstnA6k(gbl-ns-vol[medarcv~/test_results])# no vpu
bstnA6k(gbl-ns-vol[medarcv~/test_results])# ...
```

VPU Limits for Managed Volumes and Shares

Managed volumes have stricter limits on shares than direct volumes. (For the maximum shares in a direct volume, see "VPU Limits for Direct Volumes and Shares" on page 8-24.) Table 9-1 specifies managed-volume and share limits on each platform.

Table 9-1. VPU Configuration Limits

Platform	Maximum Volumes (all types)	Maximum Shares per Managed Volume	Maximum Managed Shares (total)
ARX®500 (1 VPU)	64	64	128
ARX®1000 (1 VPU)	64	64	128
ARX®6000 (per VPU)	64	64	256
• ARX®6000 with 1 ASM (2 VPUs)	128	64	512
• ARX®6000 with 2 ASMs (4 VPUs)	256	64	1,024

These limits are evaluated on a credit system; to create a new managed volume or share, its VPU must have sufficient credits. Volume limits are enforced whenever a volume is enabled, and share limits are enforced when both the share and its volume are enabled. The enable operation is denied if the VPU lacks sufficient credits.

Changing the Number of Reserved Files

Each VPU can support a limited number of files and directories in its managed volumes.

Table 9-2. Maximum Files per Platform

Platform	Maximum Files	Maximum Files per Managed Volume	Default Files per Managed Volume
ARX®500	384,000,000	128,000,000	64,000,000
ARX®1000	384,000,000	128,000,000	64,000,000
ARX®6000 (per VPU)	384,000,000	256,000,000	64,000,000
• ARX®6000 with 1 ASM (2 VPUs)	768,000,000	256,000,000	64,000,000
• ARX®6000 with 2 ASMs (4 VPUs)	1,536,000,000	256,000,000	64,000,000

You can reserve *file credits* for volumes where you expect more (or less) than the above default. From gbl-ns-vol mode, change the number of reserved files for a volume with the reserve files command.

reserve files *files*

where ***files*** (4,000,000 - 256,000,000) is the new number of files to reserve for the volume.

First, use the show VPU command to display the VPU report and list the current number of reserved files; see the File credits section of the report.

For example, the following command reserves 4,000,000 files for the '/acct' volume.

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# reserve files 4000000
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Reverting to the Default Number of Reserved Files

The no reserve files command reverts the volume to the default number of reserved files.

no reserve files

For example:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# no reserve files
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Showing All VPUs on the Current Switch

As described in the direct-volume chapter, you use the show vpu command to see all the current volume-to-VPU assignments and VPU credits. For example, the following command shows all VPUs and credits on an ARX®6000.

```
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# show vpu

Switch: bstnA6k
----------------------------------------------------------------------
VPU 1
-----
Physical Processor: 3.1
State: Normal; maximum instances
Share credits: 4 shares used (252 credits remain of total 256)
Direct share credits: 3 direct shares used (4093 credits remain of total 4096)
Volume credits: 2 volumes used (62 credits remain of total 64)
File credits: 4.0M files reserved (380M credits remain of total 384M)

Namespace          Domain  Volume                State
---------          ------  ------                -----
medco               2      /vol                  Enabled
wwmed               1      /acct                 Enabled

2 Namespaces             2 Volumes
```

```
VPU 2
-----
Physical Processor: 3.2
State: Normal; maximum instances
Share credits: 7 shares used (249 credits remain of total 256)
Direct share credits: 5 direct shares used (4091 credits remain of total 4096)
Volume credits: 6 volumes used (58 credits remain of total 64)
File credits: 132M files reserved (252M credits remain of total 384M)

Namespace         Domain  Volume               State
---------         ------  ------               -----
medarcv            1      /rcrds               Enabled
medarcv            1      /lab_equipment       Enabled
medarcv            1      /test_results        Enabled
insur              2      /claims              Enabled

2 Namespaces            4 Volumes

bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

Showing One VPU

To focus on one VPU, use the VPU number with the show vpu command. For example, the following command shows VPU 1:

```
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# show vpu 1

Switch: bstnA6k
----------------------------------------------------------------------
VPU 1
-----
Physical Processor: 3.1
State: Normal; maximum instances
Share credits: 4 shares used (252 credits remain of total 256)
Direct share credits: 3 direct shares used (4093 credits remain of total 4096)
Volume credits: 2 volumes used (62 credits remain of total 64)
```

```
File credits: 4.0M files reserved (380M credits remain of total 384M)

Namespace          Domain  Volume                  State
---------          ------  ------                  -----
medco                2     /vol                    Enabled
wwmed                1     /acct                   Enabled

2 Namespaces             2 Volumes

bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

Enabling the Volume

The final step in configuring a managed volume is to enable it. This is the same as for a direct volume: from gbl-ns-vol mode, use the enable command to enable the current volume.

The enable command starts the import of external files into all enabled shares. An import does not occur in a direct volume. This is a process of walking the directory tree on each back-end share, recording file locations, and checking against already-imported files for naming collisions.

Caution

Clients cannot directly access the filers behind this volume during or after import. Block all client access to all back-end shares before you enable the managed volume.

Note

Once the namespace's first import begins, the namespace protocol becomes more difficult to change. Before you enable the first managed volume, recall "Changing Protocols After Import" on page 7-11.

For example, the following command sequence enables the "/acct" volume in the "wwmed" namespace:

```
bstnA6k(gbl)# namespace wwmed
```

```
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# enable
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

The import happens asynchronously, so that you can issue more CLI commands while the import happens in the background. To check the progress of the import, use show namespace [status], as described below in "Monitoring the Import."

Enabling All Shares in the Volume

From gbl-ns-vol mode, you can enable all of the volume's shares with a single command. As with a direct volume, you use the enable shares command to do this (refer back to "Enabling All Shares in the Volume" on page 8-28). For example, the following command sequence enables all shares in the "/acct" volume:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# enable shares
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Taking Ownership of All Shares (optional)

Before a managed volume imports its shares, it checks the root directory in each share for a special file that marks it as "owned" by an ARX. If this marker file exists, the managed volume does not proceed with the import; no two volumes can manage the same share. You may need to override this safety mechanism for a special case.

Consider an installation that uses legacy, filer-based applications to prepare for disaster recovery: it copies all of its directories and files from a primary site to filers at another site. If an ARX manages the directories at the primary site, it places its ownership marker in the root of each share. The filer-based application copies the marker files to the remote site, along with all data files. An ARX at the backup site cannot import these shares because of the markers.

You can use the optional take-ownership flag for this special case. If the managed volume finds an ownership marker in the root of any of its shares, it overwrites the marker file. Otherwise, it imports the share as usual:

enable shares take-ownership

Caution

Do not use this option if it is possible that another ARX is managing one of the volume's shares. This would unexpectedly remove the share(s) from service at the other ARX.

The CLI prompts for confirmation before taking ownership of any shares. Enter **yes** to proceed.

For example, the following command sequence enables all shares in the "insur_bkup~/insurShdw" volume and takes ownership of all of them:

```
prtlndA1k(gbl)# namespace insur_bkup
prtlndA1k(gbl-ns[insur_bkup])# volume /insurShdw
prtlndA1k(gbl-ns-vol[insur_bkup~/insurShdw])# enable shares take-ownership
This command allows the switch to virtualize shares that are used by other Acopia switches.
Allow switch to take ownership of all the shares in this volume? [yes/no] yes
prtlndA1k(gbl-ns-vol[insur_bkup~/insurShdw])# ...
```

Disabling All Shares

Use no enable shares command to disable each of the volume's individual shares.

Caution

This is equivalent to disabling the volume, described below. This causes the volume to stop responding to clients; different client applications react to this in different ways. Some may hang, others may log errors that are invisible to the end user.

For example, this command sequence disables all of the shares in the "ns1~/vol" volume:

```
bstnA6k(gbl)# namespace ns1
bstnA6k(gbl-ns[ns1])# volume /vol
bstnA6k(gbl-ns-vol[ns1~/vol])# no enable shares
bstnA6k(gbl-ns-vol[ns1~/vol])# ...
```

Disabling the Volume

You can disable a volume to stop clients from accessing it. Just as described with a direct volume, you disable the volume with no enable in gbl-ns-vol mode. (See “Disabling the Volume” on page 8-29.)

For example, the following command sequence disables the “/acct” volume:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# no enable
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Monitoring the Import

A managed volume, unlike a direct volume, imports all back-end files and directories after you enable it. Use the show namespace status command to monitor the progress of an import:

show namespace status {*namespace* | all}

show namespace status *namespace* volume *vol-path*

show namespace status *namespace* volume *vol-path* share *share-name*

where:

namespace (1-30 characters) is the name of a namespace.

all displays details for all namespaces.

vol-path (1-1024 characters) narrows the output to a single volume.

share-name (1-1024 characters) narrows the output further, to one share.

For example, the following command shows the status of the ‘wwmed’ namespace.

```
bstnA6k(gbl)# show namespace status wwmed

Namespace: wwmed
Description: namespace for World-Wide Medical network
```

```
     Share                     Filer                                 Status
         NFS Export
    ------------------------- ------------------------------------- -----------

  Volume: /acct                                                    Enabled
     budget                    das1                                  Online
         NFS: /exports/budget

     bills                     das8                                  Online
         NFS: /work1/accting

     metadata-share            nas1                                  Online
         NFS: /vol/vol1/meta1

     bills2                    das3                                  Online
         NFS: /data/acct2

bstnA6k(gbl)# ...
```

The Status for each imported share should go through the following states:

1. Pending,
2. Importing, then
3. Online.

The show namespace command (see "Showing Namespace Details" on page 7-3) shows more detail after the "Importing" flag:

Importing: *a* items scanned, *b* items imported

Import Errors

If anything goes wrong, the failed share shows a Status of "Error" in the show namespace status output, and a specific error appears in the more-verbose show namespace output. Use show namespace (not show namespace status) to see the full error message. To correct the error, refer to the *CLI Maintenance Guide*.

Canceling a Share Import

From priv-exec mode, you can cancel the import of a single share with the cancel import command:

cancel import namespace *ns* volume *vol-path* share *share-name*

where:

ns (1-30 characters) identifies the namespace.

vol-path (1-1024 characters) is the share's volume.

share-name (1-64 characters) is the share.

For example, the following command sequence stops the import of the "wwmed~/acct~expir" share:

```
bstnA6k(gbl)# end
bstnA6k# cancel import namespace wwmed volume /acct share expir
Storage job cancelled successfully.
bstnA6k# ...
```

Reviewing the Import Report for a Multi-Protocol Volume

A multi-protocol (NFS and CIFS) volume may import successfully but still have several issues of interest. These issues center on files and/or directories with filer-generated names (FGNs, such as "dir~2") or other CIFS/NFS naming issues that cause the volume to declare them "NFS-only." An *NFS-only* file or directory, as the name implies, cannot be removed, renamed, or edited by CIFS clients. CIFS clients can only view NFS-only entries in their directory listings. NFS clients and the policy engine have full access, though rare client operations and policy migrations will incur a performance penalty. For full details, see "Finding NFS-Only Entries in a Multi-Protocol Volume" on page 8-61 of the *CLI Maintenance Guide*.

Examine the import report for each share in the multi-protocol volume. Each share has a separate import report named "import.*report-id*.*share-name*.*share-id*.rpt." Use show reports to get a list of reports, and show reports *report-name* to view the named report. Look for files or directories flagged with any Multi-Protocol-error flags. An NFS client can typically resolve these problems by renaming the file or directory through the managed volume.

For example, this shows the import report for the “shr1-old” share. The import contains several files and directories with multi-protocol issues, highlighted in bold text:

```
bstnA6k# show reports import.10.shr1-old.22.rpt
**** Share Import Report: Started at Sat Nov 18 03:26:32 2006 ****

**** Namespace: insur
**** Volume:    /claims
**** Share:     shr1-old
**** IP Addr:   192.168.25.21
**** Export:    insurance
**** Export:    /vol/vol1/NTFS-QTREE/insurance
**** Options:
****            modify:                          yes
****            rename-directories:              yes
****            rename-non-mappable-directories: yes
****            rename-files:                    yes
****            sync-attributes:                 yes
****            strict-attribute-consistency:    no

**** NOTE: Since both sync-attributes and rename-directories were specified,
****       only directories that collide with existing filenames will be
****       renamed.  Directories with colliding attributes will have their
****       attributes synchronized to the namespace.

Share                 Physical Filer
--------------------  ----------------------------------------------------------
[shr1-old          ]  192.168.25.21 NFS:/vol/vol1/NTFS-QTREE/insurance, CIFS:insurance

**** LEGEND
****
****  Actions
****   R   : Entry renamed.
****   S   : Directory attributes synchronized.
****   ?   : Directory will require “no strict-attr” to support migration (SD).
****   !   : Cannot be resolved with current 'modify' or 'no-modify' setting.
****         'modify' requires manual intervention.  'no-modify' requires 'modify'
****
****  Entry Type
****   F   : Entry is a file.
****   D   : Entry is a directory.
```

```
****
****  Issue
****   NC  : Name collision.
****   AC  : Attribute collision.
****   RA  : Attributes of share root are inconsistent.
****   MG  : Subdirectory of this share is already imported as managed share.
****   CC  : Case-blind collision (MPNS and CIFS-only).
****   ER  : Entry removed directory from filer during import.
****   AE  : Error accessing entry.
****   RV  : Reserved name on filer not imported.
****   @@  : Other error.
****
****  Multi-Protocol Issue
****   NE  : Entry found with NFS that was not found with CIFS.
****   CE  : Entry found with CIFS that was not found with NFS.
****   SL  : Symbolic link found with NFS that was not found with CIFS.
****   IC  : CIFS invalid characters found in NFS name.
****   NM  : CIFS name has characters that are not mappable to the NFS encoding.
****   FN  : A portion of the name contains a filer-generated pattern.
****   IN  : Potential name inconsistency was found.  Run 'modify' import to detect.
****   SD  : Unable to synchronize directory attributes due to a name inconsistency.


Directory Scan Phase:
================================================================================

----------------------------------------------------------
Directory Scan Start Time:   Sat Nov 18 03:26:33 2006
----------------------------------------------------------

Type      Path
--------  ----------------------------------------------------------------------
[  F NM]  /images/file012b/ (Characters: U+012b)
[R D NM]  /dir0134 -> dir0134(U+0134)_shr1-old-10
[  F IC]  /images/:3VCNC00
[  F IC]  /stats/on_the_job:2004.cnv
[  F IC]  /stats/on_the_job:2003.cnv
[  F IC]  /stats/in_home:2005/age:11-21yrs.csv
[  F IC]  /stats/in_home:2005/age:>21yrs.csv
[  F IC]  /stats/in_home:2005/age:<10yrs.csv
[  F IC]  /stats/in_home:2005/age:>21yrs.csv
[  F IC]  /stats/in_home:2005/age:<10yrs.csv
[  F IC]  /stats/in_home:2005/age:11-21yrs.csv
[  D IC]  /Claims:2001
[  D IC]  /claims:2005
[  D IC]  /:7QD4210
[  D IC]  /stats/in_home:2005
[  F CC]  /stats/piechart.ppt
[  F CC]  /stats/PieChart.ppt
[  D CC]  /tools
[  D CC]  /Tools

Directories found:                      12
Files found:                            96
Directories Scanned/Second:             12
Files Scanned/Second:                   96
Total Entries Scanned/Second:          108
Leaf Directories Not Scanned:            0

----------------------------------------------------------
Directory Scan Stop Time:    Sat Nov 18 03:26:33 2006
Directory Scan Elapsed Time: 00:00:01
----------------------------------------------------------

Directories imported by scan:                                 12
Directories imported dynamically:                              0
Files imported by scan:                                       96
Files imported dynamically:                                    0
Directories renamed due to name/attribute conflict:            1
Files renamed due to name conflict:                            0
Directory attributes synchronized:                             0
Files/directories with case-blind collisions:                  4
Files/directories with invalid CIFS characters:               10
NFS files/directories with filer generated names:              0
Files/directories found via only one protocol:                 8
CIFS files with non-mappable Unicode characters:               1

**** Elapsed time:           00:00:01
**** Share Import Report: DONE at Sat Nov 18 03:26:33 2006 ****
bstnA6k# ...
```

Showing the Volume

The direct-volume chapter discussed some show commands that focus on volumes; recall "Showing the Volume" on page 8-29. These same commands work on all volume types, including managed volumes. The difference is the output; managed volumes support policy (described in the next chapter), so any rules in the volume appear here.

To show only one volume in a namespace, add the volume clause to show namespace command. For example, the following command shows the configuration of the 'medarcv~/rcrds' volume:

```
bstnA6k# show namespace medarcv volume /rcrds

Namespace "medarcv" Configuration
Description
Metadata Cache Size: 512 MB
Proxy User: acoProxy2
SAM Reference Filer: fs2 (192.168.25.27)

Domain Information
------------------
CIFS Authentication method: Kerberos -or- NTLM Authentication Server
NTLM Authentication Servers:
  dc1
  dc1-oldStyle


Supported Protocols
-------------------
  cifs

Participating Switches
----------------------
  bstnA6k (vpu 2) [Current Switch]
```

```
Windows Management Authorization Policies
-----------------------------------------
  readOnly
  fullAccess

Volumes
-------
  /rcrds
    CIFS : compressed files: yes; named streams: yes; persistent ACLs: yes
           sparse files: yes; Unicode on disk: yes; case sensitive: no

             Volume freespace: 504GB (automatic)
              Auto Sync Files: Enabled
                Metadata size: 120k
          Metadata free space: 87GB
              Filer Subshares: Enabled; Replicate
               Oplock support: Enabled
           Notify-change mode: Normal
              CIFS path cache: Not Enabled
                        State: Enabled

                  Host Switch: bstnA6k
                     Instance: 3
                          VPU: 2 (domain 1)
                        Files: 115 used (14 dirs), 3.9M free
    Metadata shares:

      Filer          Backend Path           Contains Metadata  Status
      -------------------------------------------------------------------
      nas1           /vol/vol1/meta3        Yes                Online

    Share bulk
      Filer                        fs2 [192.168.25.27]
      CIFS Share                   bulkstorage
      Features                     cifs-acls cifs-case-blind
      CIFS Maximum Request Size   16644
      SID Translation             Yes
      Ignore SID errors           No
      Status                      Online
      Free space on storage       414GB (444,650,885,120 B)
      Transitions                 1
      Last Transition             Fri Nov  2 03:29:54 2007

    Share charts
      Filer                       fs1 [192.168.25.20]
      CIFS Share                  histories
      Features                    cifs-acls
      CIFS Maximum Request Size   16644
      SID Translation             No
      Ignore SID errors           No
      Status                      Online
      Free space on storage       15GB (16,826,933,248 B)
      Transitions                 1
      Last Transition             Fri Nov  2 03:29:50 2007

    Share rx
      Filer                       fs4 [192.168.25.29]
      CIFS Share                  prescriptions
      Features                    cifs-acls
      CIFS Maximum Request Size   16644
      SID Translation             No
      Ignore SID errors           No
      Status                      Online
      Volume Root Backing         Yes
      Critical Share              Yes
      Free space on storage       74GB (79,913,000,960 B)
      Transitions                 1
      Last Transition             Fri Nov  2 03:29:49 2007

    Share Farms
```

```
    -----------
      Share Farm medFm
        Share                      rx
        Share                      charts

        State                      Enabled
        Volume Scan Status         Complete
        Migration Status           Complete
        New File Placement Status  Enabled




    Volume Rules
    ---------------
      Rule Name                    dailyArchive
        Type                       Place Rule
        State                      Enabled
        Volume Scan Status         Complete
        Migration Status           Complete
        New File Placement Status  Enabled

bstnA6k# ...
```

Showing One Share

To show the configuration and status of one share in a managed volume, add the share clause after the volume clause. For example, the following command shows the configuration of the ‘wwmed~/acct~bills’ share:

```
bstnA6k# show namespace wwmed volume /acct share bills

Namespace “wwmed” Configuration
Description namespace for World-Wide Medical network
Metadata Cache Size: 512 MB
```

```
Domain Information
------------------



Supported Protocols
-------------------
  nfsv3

Participating Switches
----------------------
  bstnA6k (vpu 1) [Current Switch]



Volumes
-------
  /acct

            Volume freespace: 100GB (automatic)
               Metadata size: 1.5M
         Metadata free space: 32GB
           Import Protection: On
                       State: Enabled

                 Host Switch: bstnA6k
                    Instance: 2
                         VPU: 1 (domain 1)
                       Files: 4.2k used (426 dirs), 3.9M free
    Metadata shares:

      Filer          Backend Path           Contains Metadata  Status
      -------------------------------------------------------------------
      nas1           /vol/vol1/meta1        Yes                Online

    Share bills
      Filer                       das8 [192.168.25.25]
      NFS Export                  /work1/accting
      Features                    unix-perm
      Status                      Online
      Critical Share              Yes
      Free space on storage       17GB (18,803,621,888 B)
      Free files on storage       18M
      Transitions                 1
      Last Transition             Wed Apr  4 03:41:05 2007

    Share Farms
    -----------
      Share Farm fm1
        Share                     bills
        Share                     bills2
        Share                     budget

        State                     Enabled
        Volume Scan Status        Complete
        Migration Status          Complete
        New File Placement Status Enabled




    Volume Rules
    ---------------
      Rule Name                   docs2das8
        Type                      Place Rule
        State                     Enabled
        Volume Scan Status        Complete
        Migration Status          Complete
        New File Placement Status Enabled

bstnA6k# ...
```

Showing Filer Shares Behind One Volume

You can use the show namespace mapping command to show the filer shares behind a particular namespace, as described earlier in the namespace chapter. Add the volume clause to show only the shares behind that particular volume; this is the same for managed volumes as described earlier for direct volumes (see “Showing Filer Shares Behind One Volume” on page 8-34). For example, this shows the filer shares behind the “medarcv~/rcrds” volume:

```
bstnA6k# show namespace mapping medarcv volume /rcrds

Namespace                Physical Server
---------------------    ---------------------
medarcv:/rcrds
                         \\fs1\histories
                         \\fs1\prescriptions
                         \\fs2\bulkstorage
                         nas1:/vol/vol1/meta3*


Where * denotes metadata only physical server.
bstnA6k# ...
```

Showing the Volume’s Configuration

To review the configuration settings for a managed volume, identify the volume at the end of the the show global-config namespace command. This is the same syntax used for showing direct-volume configuration; recall “Showing the Volume’s Configuration” on page 8-35. The output shows all of the configuration options required to recreate the volume. The options are in order, so that they can be used as a CLI script.

For example, the following command shows the configuration for the “medarcv~/rcrds” volume:

```
bstnA6k# show global-config namespace medarcv /rcrds
;=============================== namespace ===============================
namespace medarcv
  kerberos-auth
```

```
ntlm-auth-server dc1
ntlm-auth-server dc1-oldStyle
protocol cifs
proxy-user acoProxy2
windows-mgmt-auth readOnly
windows-mgmt-auth fullAccess
sam-reference fs2
volume /rcrds
  filer-subshares replicate
  modify
  reimport-modify
  reserve files 4000000
  auto sync files
  hosted-by bstnA6k
  metadata share nas1 nfs3 /vol/vol1/meta3
  compressed-files
  named-streams
  persistent-acls
  sparse-files
  unicode-on-disk
  share bulk
    sid-translation
    filer fs2 cifs bulkstorage
    enable
    exit

  share charts
    filer fs1 cifs histories
    enable
    exit

  share rx
    critical
    filer fs4 cifs prescriptions
    enable
```

```
      exit

    share-farm medFm
      share rx
      share charts
      auto-migrate 100M
      balance Latency
      enable
      exit

    place-rule dailyArchive
      schedule hourly
      from fileset dayOld
      target share bulk
      no inline-notify
      enable
      exit

    vpu 2 domain 1
    enable
    exit

  exit

bstnA6k# ...
```

Sample - Configuring a Managed Volume

For example, this command set configures the ‘/acct’ volume on the ‘wwmed’ namespace. The ‘/acct’ volume contains two exports from two external filers:

```
bstnA6k(gbl)# namespace wwmed
This will create a new namespace.

Create namespace 'wwmed'? [yes/no] yes
bstnA6k(gbl-ns[wwmed])# volume /acct
```

```
This will create a new volume.

Create volume '/acct'? [yes/no] yes
bstnA6k(gbl-ns-vol[wwmed~/acct])# show external-filer
  Name                         IP Address     Description
  ------------------------    -------------  ----------------------------
  das1                       192.168.25.19  financial data (LINUX filer, rack 14)
  fs2                         192.168.25.27  bulk storage server (DAS, Table 3)
  fs1                         192.168.25.20  misc patient records (DAS, Table 3)
  nasE1                       192.168.25.51  NAS filer E1
  fs3                        192.168.25.28  Hematology lab server (DAS, Table 8)
  fs4                         192.168.25.29  prescription records (DAS, Table 3)
  das2                        192.168.25.22  DAS (Solaris) filer 2 (rack 16)
  das3                        192.168.25.23  DAS (Solaris) filer 3 (rack 16)
  nas1                        192.168.25.21  NAS filer 1 (rack 31)
                              192.168.25.61   (secondary)
                              192.168.25.62   (secondary)
  das7                        192.168.25.24  Redhat-LINUX filer 1
  das8                        192.168.25.25  Redhat-LINUX filer 2
  nas2                        192.168.25.44  NAS filer 2 (rack 31)
  nas3                        192.168.25.47  NAS filer 3 (rack 32)
bstnA6k(gbl-ns-vol[wwmed~/acct])# share budget
This will create a new share.

Create share 'budget'? [yes/no] yes
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~budget])# filer das1 nfs3 /exports/budget
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~budget])# enable
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~budget])# exit
bstnA6k(gbl-ns-vol[wwmed~/acct])# share bills
This will create a new share.

Create share 'bills'? [yes/no] yes
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# filer das8 nfs3 /work1/accting
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# critical
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# enable
```

```
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~bills])# exit
bstnA6k(gbl-ns-vol[wwmed~/acct])# enable
bstnA6k(gbl-ns-vol[wwmed~/acct])# exit
bstnA6k(gbl-ns[wwmed])# exit
bstnA6k(gbl)#
```

Removing a Managed Volume

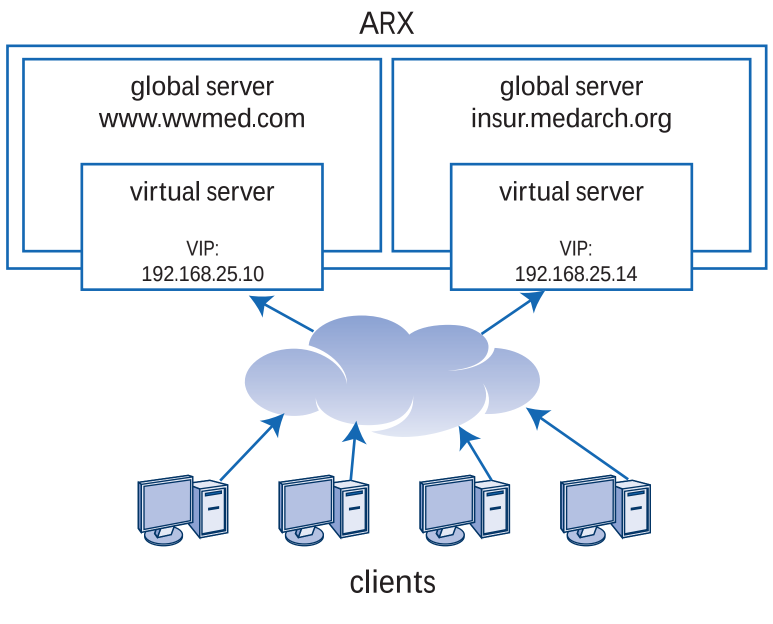
As with a direct volume, use the priv-exec remove namespace ... volume command to remove a managed volume. (Recall "Removing a Direct Volume" on page 8-37). For example, this command sequence exits to priv-exec mode and then removes the "medarcv~/lab_equipment" volume:

```
bstnA6k(gbl)# end
bstnA6k# remove namespace medarcv volume /lab_equipment

Remove volume '/lab_equipment' from namespace 'medarcv'? [yes/no] yes
Scheduling report: removeNs_medarcv_200702070957.rpt

bstnA6k# ...
```

Chapter 10

Configuring a Global Server

A *global server* is a client-entry point to the ARX's various front-end services. The global server defines a Fully-Qualified-Domain Name (FQDN; for example, "www.acopia.com") for accessing its services. A global server's services are implemented by one *virtual server* on the ARX. Each virtual server listens at a unique virtual-IP (*VIP*) address.



Concepts and Terminology

A *front-end service* is a service that is visible to clients. This is in contrast to the *back-end* filers and servers, whose services are aggregated by the ARX. A front-end service provides an interface for clients to access the aggregated back-end services. For example, the NFS and CIFS front-end services provide mount points and share names for accessing various back-end filers.

A global-server configuration includes an FQDN. This FQDN is also used as the name for any of the global server's front-end services. By convention, this is the FQDN used for client access.

Each global server contains a single *virtual server*. The virtual server is homed at a VIP address, which clients can use to access storage.

In a redundant pair, the global service and all of its components (the virtual server, its VIP, and any front-end services) fails over to the peer switch if the current switch ever fails.

Adding a Global Server

From gbl mode, use the global server command with a fully-qualified domain name (FQDN) to create a global server:

global server *fqdn*

where ***fqdn*** (1-128 characters) is the fully-qualified domain name (for example, www.company.com) to identify this service. By convention, this is the FQDN that clients will typically use to access the switch's resources.

The CLI prompts for confirmation before creating the global server; enter **yes** to proceed. This puts you into gbl-gs mode, where you must bind the server to one virtual server, set authentication parameters, and enable the global server.

For example, the following command sequence creates a global server, "www.wwmed.com:"

```
bstnA6k(gbl)# global server www.wwmed.com
This will create a new global server.
```

```
Create global server 'www.wwmed.com'? [yes/no] yes
bstnA6k(gbl-gs[www.wwmed.com])# ...
```

Setting the Windows Domain (CIFS Only)

If the global server uses back-end servers that require Windows networking, the global server needs the Windows domain to integrate with the back-end servers. Use the windows-domain command to set the Windows domain:

windows-domain *domain*

where ***domain*** can be up to 64 characters long. If CIFS clients in this domain are going to authenticate with NTLM, the namespace behind the global server must use an NTLM-authentication server in the same Windows domain. Recall "Configuring the NTLM Authentication Server" on page 3-6 (to configure the server) and "Identifying the NTLM Authentication Server" on page 7-16 (to assign the server to the namespace).

For example, the following command sequence sets the domain to "MEDARCH.ORG" for the global server at "ac1.medarch.org:"

```
bstnA6k(gbl)# global server ac1.medarch.org
bstnA6k(gbl-gs[ac1.medarch.org])# windows-domain MEDARCH.ORG
bstnA6k(gbl-gs[ac1.medarch.org])# ...
```

Setting the Pre-Windows2000 Name

Before the introduction of Windows 2000, Windows machines used short domain names, sometimes called "NT domain names," instead of the multi-domain strings in FQDNs. Today's Windows releases support both FQDNs and short-domain names. By default, the global server uses the first part of the global server's fully-qualified windows-domain name (up to 15 characters, converted to uppercase). For most installations, this is sufficient. For sites that do not conform to this naming convention, you can use the pre-win2k-name option to use a different short-domain name:

windows-domain *domain* pre-win2k-name *short-name*

where ***short-name*** (optional) can be up to 15 characters long. The CLI converts all letters to uppercase. As with the *domain*, the namespace behind this global server must have an NTLM-authentication server for the *short-name* domain, too. This is a separate NTLM-server configuration that points to the same server but supports the shortened Windows domain.

For example, the following command sequence sets the short name to “NTNET” for the global server at “ac1.medarch.org:”

```
bstnA6k(gbl)# global server ac1.medarch.org
bstnA6k(gbl-gs[ac1.medarch.org])# windows-domain MEDARCH.ORG pre-win2k-name NTNET
bstnA6k(gbl-gs[ac1.medarch.org])# ...
```

Removing the Windows Domain

Use no windows-domain to remove the Windows domain from the global server:

no windows-domain

For example:

```
bstnA6k(gbl)# global server www.nfs-only.com
bstnA6k(gbl-gs[www.nfs-only.com])# no windows-domain
bstnA6k(gbl-gs[www.nfs-only.com])# ...
```

Adding a Virtual Server

The next step in setting up a global server is creating one or more virtual servers. The virtual server listens at its VIP address for client requests; when one is received, the switch invokes one of the global server’s front-end services (NFS or CIFS) to answer the client request. A global server requires one virtual server to run its front-end services.

Use the virtual server command to create a virtual server for an ARX and assign a VIP address to the switch:

virtual server *switch-name virtual-ip-address mask* [vlan *vlan-id*]

where

switch-name (1-128 characters) is the host name of the current ARX, and

virtual-ip-address is one VIP for the switch, which you create with this command.

mask establishes the subnet part of the virtual-IP address.

vlan *vlan-id* (optional; 1-65535) is the VLAN that carries the above subnet. The default is VLAN 1.

For example, suppose the local switch has the hostname, “bstnA6k.” The following command sequence assigns it a VIP of 192.168.25.10 and adds the switch to the global server at “www.wwmed.com:”

```
bstnA6k(gbl)# global server www.wwmed.com
bstnA6k(gbl-gs[www.wwmed.com])# virtual server bstnA6k 192.168.25.10 255.255.255.0 vlan 25
bstnA6k(gbl-gs-vs[www.wwmed.com~192.168.25.10])# ...
```

This next example uses the same switch in another global server, insur.medarch.org. This global server will access the switch through another VIP, 192.168.25.14:

```
bstnA6k(gbl)# global server insur.medarch.org
bstnA6k(gbl-gs[insur.medarch.org])# virtual server bstnA6k 192.168.25.14 255.255.255.0
vlan 25
bstnA6k(gbl-gs-vs[insur.medarch.org~192.168.25.14])# ...
```

Registering with a WINS Server (CIFS)

This section only applies to virtual servers that support CIFS storage.

The Windows Internet Name Service (WINS) resolves domain-based names (such as www.mycompany.com) with IP addresses (such as 192.168.95.91). It is analogous to the Domain Name System (DNS), except that WINS integrates with the dynamic-addressing protocol, Dynamic Host Configuration Protocol (DHCP).

If you identify a WINS server for this network, the virtual server registers its NetBIOS name with the WINS server. This makes it possible for other WINS clients to find the virtual server on this switch. Use the wins command to identify a WINS server:

wins *ip-address*

where ***ip-address*** is the address of the name server.

If the WINS server supports multi-byte character encoding, set the proper character encoding at the namespace behind this virtual server. Refer back to "Setting CIFS Character Encoding" on page 7-13.

For example, this command sequence configures a WINS server for the virtual server at VIP 192.168.25.15:

```
bstnA6k(gbl)# global server ac1.medarch.org
bstnA6k(gbl-gs[ac1.medarch.org])# virtual server bstnA6k 192.168.25.15 255.255.255.0 vlan
25
bstnA6k(gbl-gs-vs[ac1.medarch.org~192.168.25.15])# wins 192.168.25.20
bstnA6k(gbl-gs-vs[ac1.medarch.org~192.168.25.15])# ...
```

Removing the WINS-Server Setting

You can stop the virtual server from registering its NetBIOS name with a WINS server. This is only effective if you do it before you enable the virtual server (below). Use the no form of the wins command to remove the IP address of the WINS server:

no wins

Note

The virtual server answers broadcast requests for its shares, whether it is registered with a WINS server or not.

For example:

```
bstnA6k(gbl)# global server ac1.medarch.org
bstnA6k(gbl-gs[ac1.medarch.org])# virtual server bstnA6k 192.168.25.15 255.255.255.0 vlan
25
bstnA6k(gbl-gs-vs[ac1.medarch.org~192.168.25.15])# no wins
bstnA6k(gbl-gs-vs[ac1.medarch.org~192.168.25.15])# ...
```

Setting the NetBIOS Name (optional, CIFS)

This section only applies to virtual servers that support CIFS storage.

The virtual server's NetBIOS name is the server name that appears in Windows network browsers. This appears in the "Server Name" column when you issue a Windows **net view** command. For example:

```
U:\>net view
Server Name           Remark
-------------------------------------------------------------------------------
\\PERSONNEL
\\ARCHIVE1          A full tree of 5-year-old records
```

The above net view displays two CIFS servers, PERSONNEL and ARCHIVE1.

The default NetBIOS name for the virtual server is the first component of the global server's FQDN; for example, "FS" for the global server at "fs.nt.org." If that component is longer than 15 bytes, only the first 15 bytes are used.

Use the optional wins-name command to reset the name:

wins-name *netbios-name*

where ***netbios-name*** (1-15 bytes) is the NetBIOS name to be advertised to the WINS server. The first character must be a letter, and the remaining characters can be letters, numbers, or underscores (_). If you are using Latin characters (including ASCII), each character is one byte, so the name can be up to 15 characters. Each character may be more than one byte for non-Latin characters, limiting you to a smaller number of characters.

For example, this command sequence resets the NetBIOS name to "INSURANCE" for the virtual server at 192.168.25.14:

```
bstnA6k(gbl)# global server fs.nt.org
bstnA6k(gbl-gs[fs.nt.org])# virtual server bstnA6k 192.168.25.14 255.255.255.0 vlan 25
bstnA6k(gbl-gs-vs[fs.nt.org~192.168.25.14])# wins-name INSURANCE
bstnA6k(gbl-gs-vs[fs.nt.org~192.168.25.14])# ...
```

Adding a NetBIOS Alias

Some installations use multiple NetBIOS names for a single CIFS server. To mimic this configuration, use the wins-alias command (in gbl-gs-vs mode) for each additional NetBIOS name:

wins-alias *netbios-alias*

where ***netbios-alias*** (1-15 bytes) is a NetBIOS alias to be advertised to the WINS server. The first character must be a letter, and the remaining characters can be letters, numbers, or underscores (_). The same character/byte limitations apply to this name as the NetBIOS name itself (see above).

For example, this command sequence adds a NetBIOS alias to the “192.168.25.12” server:

```
bstnA6k(gbl)# global server fs.nt.org
bstnA6k(gbl-gs[fs.nt.org])# virtual server bstnA6k 192.168.25.12 255.255.255.0 vlan 25
bstnA6k(gbl-gs-vs[fs.nt.org~192.168.25.12])# wins-alias HR_DISK
bstnA6k(gbl-gs-vs[fs.nt.org~192.168.25.12])# ...
```

Removing NetBIOS Aliases

To remove one NetBIOS alias (or all of them) from the virtual server, use no wins-alias:

no wins-alias [*netbios-name*]

where ***netbios-name*** (optional, 1-15 characters) identifies the alias to remove. If you do not specify a particular NetBIOS alias, this command removes all of them.

The CLI prompts for confirmation if you choose to remove all aliases. Enter **yes** to continue.

For example, this command sequence removes all NetBIOS aliases from the virtual server at 192.168.25.15:

```
bstnA6k(gbl)# global server ac1.medarch.org
bstnA6k(gbl-gs[ac1.medarch.org])# virtual server bstnA6k 192.168.25.15 255.255.255.0 vlan 
25
bstnA6k(gbl-gs-vs[ac1.medarch.org~192.168.25.15])# no wins-alias
Delete all NETBIOS aliases for this Virtual Server? [yes/no] yes
bstnA6k(gbl-gs-vs[ac1.medarch.org~192.168.25.15])# ...
```

Reverting to the Default NetBIOS Name

You can revert to the default NetBIOS name with the no wins-name command. The default NetBIOS name is the first component of the global server's FQDN (for example, "\\FTP1" for global server "ftp1.government.gov").

no wins-name

The CLI prompts for confirmation before deleting the name.

For example, the following command sequence sets the NetBIOS name back to its default ("\\MYCO") for the virtual server at "192.168.25.87:"

```
bstnA6k(gbl)# global server myco.com
bstnA6k(gbl-gs[myco.com])# virtual server bstnA6k 192.168.25.87 255.255.255.0 vlan 25
bstnA6k(gbl-gs-vs[myco.com~192.168.25.87])# no wins-name
Delete all NETBIOS aliases for this Virtual Server? [yes/no] yes
bstnA6k(gbl-gs-vs[myco.com~192.168.25.87])# ...
```

Enabling a Virtual Server

The final step in virtual-server configuration is to enable it. You must explicitly enable a virtual server for the switch to accept clients at its VIP address. From gbl-gs-vs mode, use the enable command to activate the virtual server:

enable

This makes it possible for global-server clients to access the virtual server's services. (You must also enable the global server, so that it accepts clients to pass to the virtual server.)

For example, the following command sequence enables the virtual server on "bstnA6k" at 192.168.25.10, and then enables its global server at www.wwmed.com:

```
bstnA6k(gbl)# global server www.wwmed.com
bstnA6k(gbl-gs[www.wwmed.com])# virtual server bstnA6k 192.168.25.10 255.255.255.0 vlan 25
bstnA6k(gbl-gs-vs[www.wwmed.com~192.168.25.10])# enable
bstnA6k(gbl-gs-vs[www.wwmed.com~192.168.25.10])# exit
bstnA6k(gbl-gs[www.wwmed.com])# enable
bstnA6k(gbl-gs[www.wwmed.com])# ...
```

Disabling a Virtual Server

Disabling a virtual server makes it impossible for clients to access the particular switch's front-end services (such as CIFS or NFS) through that virtual server's IP address. Use no enable in gbl-gs-vs mode to disable a virtual server.

no enable

This command gracefully terminates all client connections at the VIP address, allowing current transactions and sessions to finish while blocking any new connections.

For example, the following command sequence disables a virtual server:

```
bstnA6k(gbl)# global server www.wwmed.com
bstnA6k(gbl-gs[www.wwmed.com])# virtual server bstnA6k 192.168.25.11 255.255.255.0 vlan 25
bstnA6k(gbl-gs-vs[www.wwmed.com~192.168.25.11])# no enable
bstnA6k(gbl-gs-vs[www.wwmed.com~192.168.25.11])# ...
```

Removing a Virtual Server

Once you remove a virtual server, the global server's front-end services stop running on the virtual server's ARX.

Use the no form of the virtual server command to remove a virtual server:

no virtual server *switch-name virtual-ip-address*

where

switch-name (1-128 characters) is the host name of the ARX, and

virtual-ip-address identifies the virtual server.

Before removing the virtual server, the CLI prompts for confirmation. Enter **yes** to continue.

For example, the following command sequence removes a virtual server from global server www.wwmed.com:

```
bstnA6k(gbl)# global server www.wwmed.com
bstnA6k(gbl-gs[www.wwmed.com])# no virtual server bstnA6k 192.168.123.9
Delete virtual server ''192.168.123.9'' on switch ''bstnA6k''? [yes/no] yes
bstnA6k(gbl-gs[www.wwmed.com])# ...
```

Enabling the Global Server

The final step in global-server configuration is to enable it. Use the enable command to activate the global server:

enable

For example, the following command sequence enables the global server at "www.wwmed.com:"

```
bstnA6k(gbl)# global server www.wwmed.com
bstnA6k(gbl-gs[www.wwmed.com])# enable
bstnA6k(gbl-gs[www.wwmed.com])# ...
```

Disabling the Global Server

Disabling the global server makes it impossible for clients to access any of its resources. Use no enable in gbl-gs mode to disable the global server.

no enable

This command gracefully terminates all client connections to the global server, allowing current transactions and sessions to finish while blocking any new connections.

For example, the following command sequence disables the "www.wwmed.com" global server:

```
bstnA6k(gbl)# global server www.wwmed.com
bstnA6k(gbl-gs[www.wwmed.com])# no enable
bstnA6k(gbl-gs[www.wwmed.com])# ...
```

Showing All Global Servers

Use the show global server command to show all global servers on the ARX:

show global server

For example:

```
bstnA6k(gbl)# show global server
```

```
Domain Name             State                Windows Domain
---------------------------------------------------------------------------
ac1.medarch.org         Enabled              MEDARCH.ORG (NTNET)

  Switch                State    VIP                  VLAN  VMAC
                                 WINS Server                WINS Name
  -------------------------------------------------------------------------
  bstnA6k               Enabled  192.168.25.15/24      25   00:0a:49:00:0a:c0
                                 192.168.25.20              (none)

  Description
  -------------------------------------------------------------------------
  CIFS server for hospital-net storage


Domain Name             State                Windows Domain
---------------------------------------------------------------------------
acopiaFiler             Enabled

  Switch                State    VIP                  VLAN  VMAC
                                 WINS Server                WINS Name
  -------------------------------------------------------------------------
  bstnA6k               Enabled  192.168.25.12/24      25   00:0a:49:00:0a:c0
                                 (none)                     (none)

  Description
  -------------------------------------------------------------------------



Domain Name             State                Windows Domain
---------------------------------------------------------------------------
insur.medarch.org       Enabled              MEDARCH.ORG

  Switch                State    VIP                  VLAN  VMAC
```

```
                               WINS Server              WINS Name
  -------------------------------------------------------------------------
  bstnA6k              Enabled  192.168.25.14/24    25   00:0a:49:00:0a:c0
                                192.168.25.20            INSURANCE

  Description
  -------------------------------------------------------------------------
  CIFS and NFS server for hospital insurance claims


Domain Name              State                 Windows Domain
---------------------------------------------------------------------------
www.wwmed.com            Enabled

  Switch               State    VIP                VLAN  VMAC
                                WINS Server              WINS Name
  -------------------------------------------------------------------------
  bstnA6k              Enabled  192.168.25.10/24    25   00:0a:49:00:0a:c0
                                (none)                   (none)

  Description
  -------------------------------------------------------------------------
  global NFS server for network hospitals

bstnA6k(gbl)# ...
```

Showing One Global-Server

To see one global server, identify a particular global server with the show global server command:

show global server *fqdn*

where ***fqdn*** (1-128 characters) is the fully-qualified domain name (for example, www.company.com) for the global server.

For example, the following command shows the global server at “www.wwmed.com:”

```
bstnA6k(gbl)# show global server www.wwmed.com

Domain Name               State                  Windows Domain
---------------------------------------------------------------------------
www.wwmed.com             Enabled

  Switch                  State     VIP                  VLAN  VMAC
                                    WINS Server                WINS Name
  -------------------------------------------------------------------------
  bstnA6k                 Enabled   192.168.25.10/24      25   00:0a:49:00:08:c0
                                    (none)                     (none)

  Description
  -------------------------------------------------------------------------
  global NFS server for network hospitals

bstnA6k(gbl)# ...
```

Removing a Global Server

You cannot remove the global server if any front-end services reference it, or if it is enabled. After you remove all services and disable the global server, use the no form of the global server command to remove it:

no global server *fqdn*

where ***fqdn*** (1-128 characters) is the fully-qualified domain name (for example, www.company.com) for the global server.

The CLI prompts for confirmation before deleting the global server; enter **yes** to continue. For example, the following command sequence removes global server ftp.medarch.org:

```
bstnA6k(gbl)# global server ftp.medarch.org
bstnA6k(gbl-gs[ftp.medarch.org])# no enable
bstnA6k(gbl-gs[ftp.medarch.org])# exit
bstnA6k(gbl)# no global server ftp.medarch.org
Delete global server 'ftp.medarch.org'? [yes/no] yes
```

```
bstnA6k(gbl)# ...
```

Sample - Configuring a Global Server

The following command sequence sets up a global server for www.wwmed.com.

Create the global server:

```
bstnA6k(gbl)# global server ac1.medarch.org
This will create a new global server.

Create global server 'ac1.medarch.org'? [yes/no] yes
```

Join a Windows domain, MEDARCH.ORG:

```
bstnA6k(gbl-gs[ac1.medarch.org])# windows-domain MEDARCH.ORG
```

Bind to the current ARX, “bstnA6k.” Assign a VIP address, 192.168.25.15, to the server:

```
bstnA6k(gbl-gs[ac1.medarch.org])# virtual server bstnA6k 192.168.25.15 255.255.255.0 vlan
25
```

Identify the local WINS server:

```
bstnA6k(gbl-gs[ac1.medarch.org])# wins 192.168.25.20
```

Enable and exit both the virtual server and the global server:

```
bstnA6k(gbl-gs-vs[ac1.medarch.org~192.168.25.15])# enable
bstnA6k(gbl-gs-vs[ac1.medarch.org~192.168.25.15])# exit
bstnA6k(gbl-gs[ac1.medarch.org])# enable
bstnA6k(gbl-gs[ac1.medarch.org])# exit
bstnA6k(gbl)#
```

Next

For a global server to host any back-end services (such as filer storage), you must configure one or more front-end services. The front-end services provide access to namespace storage; the virtual server(s) run the front-end services on behalf of the global server. Chapter 11, *Configuring Front-End Services*, describes how to configure various front-end services for a global server.

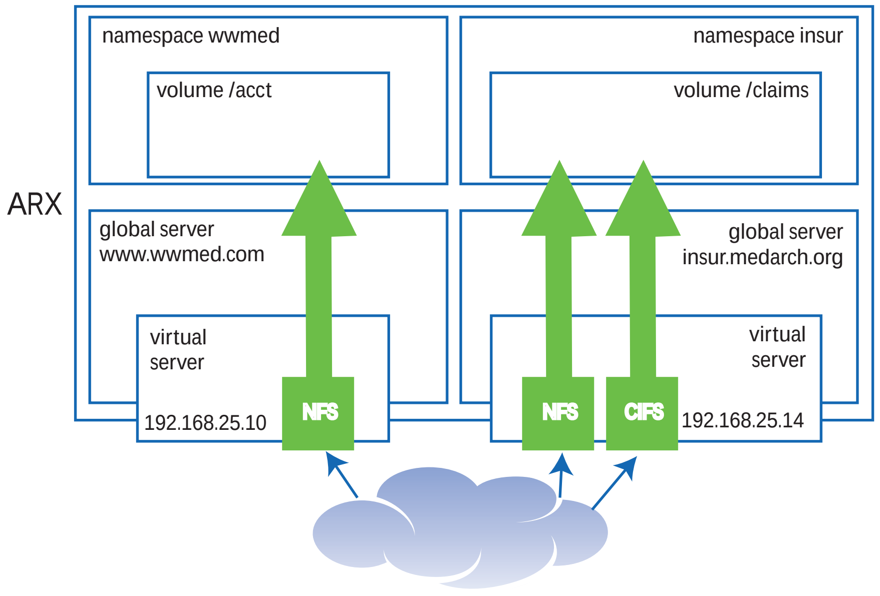
Chapter 11

Configuring Front-End Services

Front-end services provide client access to namespace storage. Supported front-end services include

- Network File System (NFS), and
- Common Internet File System (CIFS).

You can enable one or more of these services on a global server, so that clients can access them through the global server's fully-qualified domain name (FQDN) or the virtual server's VIP. For each service, you determine which namespace volumes are available as storage resources.



Before You Begin

To offer any front-end services, you must first

- add one or more NAS filers, as described in Chapter 6, *Adding an External Filer*;
- create at least one namespace as a storage resource, as described in Chapter 7, *Configuring a Namespace*;
- create at least one volume in the namespace, a direct volume (recall Chapter 8, *Adding a Direct Volume*) or a managed volume (see Chapter 9, *Adding a Managed Volume*);

  and
- configure a global server that uses this ARX as a virtual server, as described in Chapter 10, *Configuring a Global Server*.

Configuring NFS

From gbl mode, use the nfs command to instantiate NFS service for a global server:

nfs *fqdn*

where ***fqdn*** (1-128 characters) is the fully-qualified domain name (for example, “www.organization.org”) for the global server.

The CLI prompts for confirmation before creating the service; enter **yes** to proceed. (You can disable all such create-confirmation prompts with the terminal expert command.) This places you in gbl-nfs mode, from which you can export various namespace volumes.

For example, the following command sequence creates an NFS service at www.wwmed.com:

```
bstnA6k(gbl)# nfs www.wwmed.com
This will create a new NFS service.


Create NFS service 'www.wwmed.com'? [yes/no] yes
bstnA6k(gbl-nfs[www.wwmed.com])# ...
```

From gbl-nfs mode, you must export at least one namespace volume and then enable the NFS service, as described in the following subsections.

Exporting a Namespace Volume

If a namespace volume is configured for NFS, you can offer it as an NFS export through a global server. Each NFS service can support volumes from a single namespace.

Use the export command to offer a namespace volume through the current NFS service:

export *namespace vol-path* [as *export-path*] [access-list *list-name*]

where

namespace (1-30 characters) can be any namespace that supports NFS,

vol-path (1-1024 characters) is the path to one of the namespace's volumes (for example, "/etc") or volume sub paths ("/etc/init.d"), and

as *export-path* (optional, 1-1024 characters) sets an optional, advertised path to the volume. If entered, NFS clients see this path as an available export instead of the ***volume*** path.

access-list *list-name* (optional, 1-64 characters) uses an access list to control which IP subnets can and cannot access this NFS service. See "Adding an NFS Access List" on page 4-9 for instructions on configuring an NFS access list. If you omit this option, clients from *any* subnet can access the export with read/write permission, root-squash enabled, and root is squashed to UID 65534 and GID 65534.

Whether or not you use an access list, non-root UIDs are passed through to the back-end filer for possible authentication at the filer.

Use show global-config namespace to see all available namespaces and volumes on the ARX. Each NFS service can support volumes from *one* namespace only. Choose only volumes that support NFS.

For example, the following command sequence adds an NFS export to the NFS service at www.wwmed.com:

```
bstnA6k(gbl)# nfs www.wwmed.com
```

```
bstnA6k(gbl-nfs[www.wwmed.com])# show global-config namespace
;=============================== namespace ===============================
...

;=============================== namespace ===============================
namespace wwmed
  description "namespace for World-Wide Medical network"
  protocol nfs3
  volume /acct
    import protection
    metadata critical
...
bstnA6k(gbl-nfs[www.wwmed.com])# export wwmed /acct access-list eastcoast
bstnA6k(gbl-nfs[www.wwmed.com])# ...
```

Stopping an NFS Export

Use the no form of the export command to stop NFS access to a namespace volume:

no export *namespace vol-path*

where

namespace (1-30 characters) is the namespace containing the NFS export, and

vol-path (1-1024 characters) is the path to a current NFS export (for example, "/acct").

For example, the following command sequence stops the NFS server at www.wwmed.com from exporting "wwmed /test":

```
bstnA6k(gbl)# nfs www.wwmed.com
bstnA6k(gbl-nfs[www.wwmed.com])# no export wwmed /test
bstnA6k(gbl-nfs[www.wwmed.com])# ...
```

Disabling NLM (optional)

The NFS service implements the NFS Lock Manager (NLM) protocol. NLM is a voluntary protocol that NFS-client applications can use to write-protect a file or file region. NFS client A can use NLM to *lock* a region of a file; if clients B and C are also NLM-compliant, they will not write to that region until client A releases the lock.

While the NFS service is disabled, you have the option to disable NLM.

Note

This requires careful consideration for an NFS service in front of a direct volume (direct volumes were described in Chapter 8, *Adding a Direct Volume*). A direct volume can offer NLM file locks to its clients, but clients that access the back-end filers directly (not through the ARX) or through another direct volume are unaware of those locks. Therefore, direct volumes should *not* offer NLM unless you are sure that all client access goes through this volume.

If multiple NFS services export the same volume, consistently enable or disable NLM for all of them. This applies to managed volumes as well as direct volumes.

Use the no nlm enable command from gbl-nfs mode to disable NLM at this NFS service:

no nlm enable

This prevents the front-end service from answering any NLM requests for file locks; the CLI prompts for confirmation before doing this. Enter **yes** to continue.

For example:

```
bstnA6k(gbl)# nfs acopiaFiler
bstnA6k(gbl-nfs[acopiaFiler])# no nlm enable
Disable Network Lock Manager on acopiaFiler? [yes/no] yes
bstnA6k(gbl-nfs[acopiaFiler])# ...
```

For commands to view current NLM locks and NLM statistics, refer to the Front-End Services chapter in the *CLI Reference Guide*.

Enabling NLM

While the NFS service is disabled, you can use nlm enable to re-enable NLM processing:

nlm enable

This causes the front-end service to answer all NLM requests.

For example:

```
bstnA6k(gbl)# nfs www.wwmed.com
bstnA6k(gbl-nfs[www.wwmed.com])# nlm enable
bstnA6k(gbl-nfs[www.wwmed.com])# ...
```

Enabling NFS Service

The final step in NFS configuration is to enable it. Use the enable command from gbl-nfs mode to activate the NFS service:

enable

This makes all declared NFS exports available to clients at the global server's VIP.

For example, the following command sequence enables NFS for the global server at "www.wwmed.com:"

```
bstnA6k(gbl)# nfs www.wwmed.com
bstnA6k(gbl-nfs[www.wwmed.com])# enable
bstnA6k(gbl-nfs[www.wwmed.com])# ...
```

Disabling NFS

Disabling NFS stops all NFS connections to the global server. Use no enable in gbl-nfs mode to disable NFS.

no enable

For example, the following command sequence disables NFS for the "www.wwmed.com" global server:

```
bstnA6k(gbl)# nfs www.wwmed.com
```

```
bstnA6k(gbl-nfs[www.wwmed.com])# no enable
bstnA6k(gbl-nfs[www.wwmed.com])# ...
```

Notifications to NLM Clients

As described above, the NFS service can implement the NFS Lock Manager (NLM) protocol. If you used no nlm enable to stop NLM, skip to the next section.

In an NFS service where NLM is enabled, a no enable followed by an enable triggers a notification to NLM clients. When the NFS service comes back up, it follows the NLM protocol for server recovery from a crash. The NFS service sends an SM_NOTIFY message to all NLM clients with file locks. According to the protocol, all clients should then reclaim their locks.

For commands to view current NLM locks and NLM statistics, refer to the Front-End Services chapter in the *CLI Reference Guide*.

Listing All NFS Services

Use the show nfs-service command to see summaries for all NFS services:

show nfs-service

For example:

```
bstnA6k(gbl)# show nfs-service

Domain Name                 Namespace         Description
------------------------------------------------------------------------------
www.wwmed.com               wwmed             mount point for wwmed storage
acopiaFiler                 medco
insur.medarch.org           insur             insurance records for WW Medical

bstnA6k(gbl)# ...
```

Showing One NFS Service

Identify a particular FQDN with the show nfs-service command to focus on one NFS service:

show nfs-service *fqdn*

where ***fqdn*** (1-128 characters) is the fully-qualified domain name (for example, www.company.com) for the global server.

This shows detailed configuration information for the service.

For example, the following command shows the NFS configuration for the global server at “www.wwmed.com:”

```
bstnA6k(gbl)# show nfs-service www.wwmed.com

Virtual Server: www.wwmed.com
Namespace:      wwmed
Description:    mount point for wwmed storage

NFS State:      Enabled
NLM State:      Enabled

Exports:

  Directory             Export As               State       Access
  ------------------------------------------------------------------------
  /acct                 /acct                   Online      eastcoast
bstnA6k(gbl)# ...
```

Showing Details for All NFS Services

To show details for all NFS front-end services, use show nfs-service all:

show nfs-service all

This shows the detailed view of all configured NFS services.

Sample - Configuring an NFS Front-End Service

The following command sequence sets up NFS service on a global server called “www.wwmed.com:”

```
bstnA6k(gbl)# nfs www.wwmed.com
bstnA6k(gbl-nfs[www.wwmed.com])# show global-config namespace wwmed
;=============================== namespace ================================
namespace wwmed
  description “namespace for World-Wide Medical network”
  protocol nfs3
  volume /acct
    import protection
...
bstnA6k(gbl-nfs[www.wwmed.com])# export wwmed /acct access-list eastcoast
bstnA6k(gbl-nfs[www.wwmed.com])# enable
bstnA6k(gbl-nfs[www.wwmed.com])# exit
bstnA6k(gbl)# show nfs-service www.wwmed.com

Virtual Server: www.wwmed.com
Namespace:      wwmed
Description:

NFS State:      Enabled
NLM State:      Enabled

Exports:

  Directory              Export As                State       Access
  -------------------------------------------------------------------------
  /acct                  /acct                    Online      eastcoast
bstnA6k(gbl)# ...
```

Removing an NFS Service

You can remove an NFS service from a global server to both disable the service and remove its configuration. Use the no form of the nfs command to remove an NFS-service configuration from a global server:

no nfs *fqdn*

where ***fqdn*** (1-128 characters) is the fully-qualified domain name (for example, “www.organization.org”) for the service’s global server.

The CLI prompts for confirmation before removing the service; enter **yes** to proceed.

For example, the following command sequence removes the NFS-service offering for athos.nfstest.net:

```
bstnA6k(gbl)# no nfs athos.nfstest.net
Delete NFS service on 'athos.nfstest.net'? [yes/no] yes
bstnA6k(gbl)#
```

Changing the NFS/TCP Timeout Behavior

When an NFS/TCP connection to a back-end share times out, the NFS service (by default) disconnects its front-end client. You can change this default: instead of disconnecting from the client, the NFS service can send back an NFS I/O error (NFSERR_IO or NFS3ERR_IO). With the same command, you can reduce the default time-out period of 105 seconds. From gbl mode, use the nfs tcp timeout command to make these changes:

nfs tcp timeout *seconds*

where ***seconds*** (5-104) is the time-out period for NFS/TCP connections.

For example, this command sets the timeout to 30 seconds (and stops disconnecting clients on timeout):

```
bstnA6k(gbl)# nfs tcp timeout 30
bstnA6k(gbl)# ...
```

Showing the NFS/TCP Timeout

Use the show nfs tcp command to view the current client-connection behavior and timeout period for NFS/TCP timeouts:

show nfs tcp

For example, this system has the behavior configured above:

```
bstnA6k(gbl)# show nfs tcp


Transaction Timeout

  Behavior:    Return I/O Error

  Inactivity:  30 seconds

bstnA6k(gbl)# ...
```

Reverting to the Default Timeout and Behavior

By default, an NFS/TCP connection to any filer times out in 105 seconds, and the NFS service drops the connection to the client on timeout. Use no nfs tcp timeout to return to this default for all NFS filers and services:

no nfs tcp timeout

For example:

```
bstnA6k(gbl)# no nfs tcp timeout

bstnA6k(gbl)# ...
```

Configuring CIFS

From gbl mode, use the cifs command to instantiate CIFS service for a global server:

cifs *fqdn*

where ***fqdn*** (1-128 characters) is the fully-qualified domain name for the global server (for example, “myserver.organization.org”). If this CIFS service runs in an Active Directory forest and/or uses Kerberos to authenticate clients, this must be in a domain in the Active-Directory forest (configured in “Adding an Active-Directory Forest (Kerberos)” on page 3-10).

The CLI prompts for confirmation before creating the service; enter **yes** to proceed. (You can disable all such create-confirmation prompts with the terminal expert command.) This places you in gbl-cifs mode, from which you can share various namespace volumes.

For example, the following command sequence offers CIFS service at ac1.medarch.org:

```
bstnA6k(gbl)# cifs ac1.medarch.org
This will create a new CIFS service.

Create CIFS service 'ac1.medarch.org'? [yes/no] yes
bstnA6k(gbl-cifs[ac1.medarch.org])# ...
```

From gbl-cifs mode, you must export at least one namespace volume and then enable the CIFS service, as described in the following subsections.

Sharing a Namespace Volume

If a namespace volume is configured for CIFS, you can offer it as a share through the CIFS-service configuration. You can offer as many CIFS shares as desired from a given namespace; each CIFS service supports a single namespace.

Use the export command to share a namespace volume through the current CIFS service:

> **export *namespace vol-path* [as *share-name*] [description *description*]**
>
> where
>
> > ***namespace*** (1-30 characters) can be any namespace that supports CIFS.
> >
> > ***vol-path*** (1-1024 characters) is the path to one of the namespace's volumes (for example, "/oneVol") or volume sub paths ("oneVol/apps/myApps").
> >
> > **as *share-name*** (optional; 1-1024 characters) sets a share name for the volume. Potential CIFS clients see the ***share-name*** as an available share when, for example, they issue the **net view** command from a DOS prompt. If you omit this, the share name defaults to the volume name without the leading slash (/). By repeating this command with other share names, you can share the same volume under multiple names.
> >
> > **description *description*** (optional; 1-64 characters) sets an optional description for the CIFS share. This describes the share in a Windows network browser (for example, **net view**). If the description includes any spaces, it must be surrounded by quotation marks ("").

Use show global-config namespace to see all available namespaces on the ARX. The CIFS service supports volumes from a single CIFS namespace.

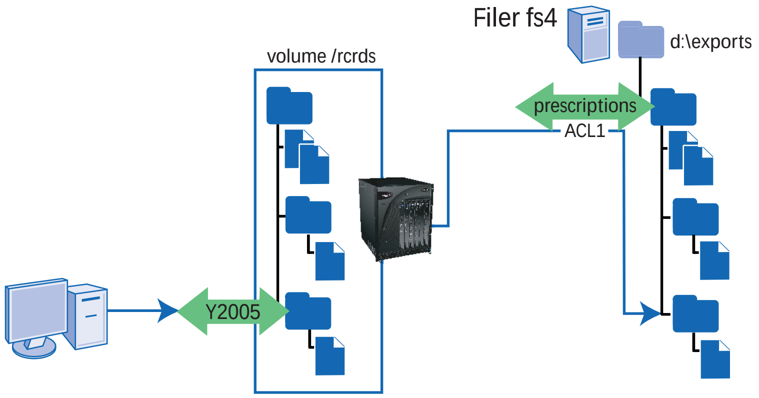
For example, the following command sequence adds a CIFS share to the CIFS service at ac1.medarch.org:

```
bstnA6k(gbl)# cifs ac1.medarch.org
bstnA6k(gbl-cifs[ac1.medarch.org])# show global-config namespace
;================================ namespace ================================
namespace medco
  protocol nfs3tcp
...
;================================ namespace ================================
namespace medarcv
  kerberos-auth
  ntlm-auth-server dc1
  ntlm-auth-server dc1-oldStyle
  protocol cifs
  proxy-user acoProxy2
  windows-mgmt-auth readOnly
  windows-mgmt-auth fullAccess
  sam-reference fs2
  volume /lab_equipment
...
    enable
    exit

  volume /rcrds
    filer-subshares replicate
    modify
...
bstnA6k(gbl-cifs[ac1.medarch.org])# export medarcv /rcrds as ARCHIVES
description "2-year-old medical records"
bstnA6k(gbl-cifs[ac1.medarch.org])# ...
```

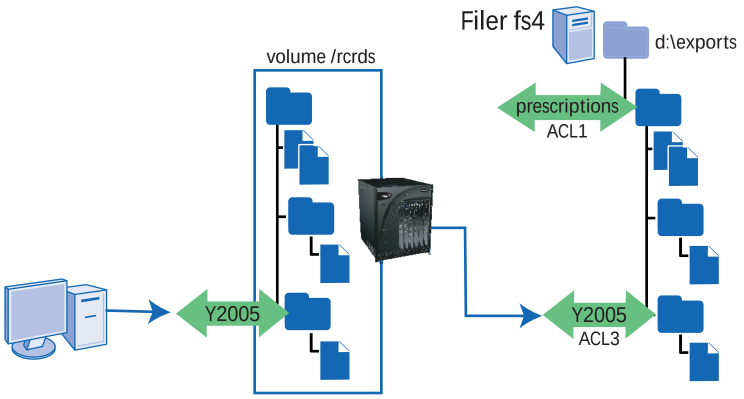
Exporting a Filer Subshare (and Using its ACL)

This section only applies to managed volumes. Skip all of the “subshare” sections if you are sharing a direct volume.

The CIFS service accesses each back-end share through its root, whether or not you export a directory below the root of the volume. For example, suppose you export “/rcrds/2005” as a *subshare* of the above “/rcrds” share. When a client accesses that subshare, the CIFS service goes through the top-level share at the back-end filer (for example, “\\fs4\prescriptions”) and drops down to the /2005 directory. This bypasses any share-level ACL that the filer may have for its own version of the /2005 subshare.



You can configure the CIFS service to pass CIFS clients through to the same subshares, thus ensuring that they use the correct share-level ACL:



The volume and filers must be properly prepared before your CIFS service can offer this subshare service. A subshare must have the same name, ACL, and position in the directory tree (relative to the share root) on every filer behind the volume. To continue the example, every filer share behind the “medarcv~/rcrds” volume must have

- a /2005 directory under its root,
- the same share name for it (“Y2005” in the above illustration), and
- the same ACL (“ACL3,” or its equivalent).

The managed-volume chapter discussed this preparation. Refer back to “Supporting Subshares and their ACLs” on page 9-20, and “Replicating Subshares at all of the Volume’s Filers” on page 9-23.

With the volume properly prepared, you can use the filer-subshare flag in the export command:

export *namespace vol-path* as *subshare-name* filer-subshare [description *description*]

where

namespace and

vol-path are both described above.

subshare-name must match the subshare name used at all the back-end filers.

filer-subshare specifies that clients should pass through to the back-end subshare.

description *description* is also described above.

For example, the following command sequence adds the “Y2005” subshare to the CIFS service at ac1.medarch.org:

```
bstnA6k(gbl)# cifs ac1.medarch.org
bstnA6k(gbl-cifs[ac1.medarch.org])# export medarcv /rcrds/2005 as Y2005 filer-subshare
bstnA6k(gbl-cifs[ac1.medarch.org])# ...
```

Exposing Hidden Subshares

Some filer subshares can be hidden by having a dollar sign ($) at the ends of their share names (for example, “myshare$”). Most views of the filer’s CIFS shares do not show these names. The CIFS front-end service can expose a hidden subshare by using a slightly-different name for its front-end subshare; the back-end name without the “$” (such as “myshare”).

To expose hidden subshares, use the hidden keyword after filer-subshare:

> **export *namespace vol-path* as *subshare-name* filer-subshare hidden [description *description*]**
>
> where
>
> > ***subshare-name*** is the name of the subshare in the CIFS service only (for example, “subshare1”). At the back-end filer(s), the same subshare name should already be configured with “$” at the end (“subshare1$”).
> >
> > **hidden** is required.

For example, this command sequence adds the “old$” subshare and exports it under the exposed-share name, “old,” from the “ac1.medarch.old” service:

```
bstnA6k(gbl)# cifs ac1.medarch.org
bstnA6k(gbl-cifs[ac1.medarch.org])# export medarcv /rcrds/old as old filer-subshare hidden
bstnA6k(gbl-cifs[ac1.medarch.org])# ...
```

Exporting all Filer Subshares at Once

This section only applies to managed volumes. Skip all of the “subshare” sections if you are sharing a direct volume.

You can use a single command to export multiple filer subshares. This presumes that the volume is prepared with multiple subshares, perhaps through subshare replication (recall “Replicating Subshares at all of the Volume’s Filers” on page 9-23). Exit to priv-exec mode and use the cifs export-subshares command:

cifs export-subshares *namespace volume fqdn* [tentative]

where

namespace (1-30 characters) can be any namespace that supports CIFS.

volume (1-1024 characters) identifies the root of one volume (for example, “/oneVol”) with CIFS subshares.

fqdn (1-255 characters) is an existing CIFS service. The subshares will be available to clients through this service.

tentative (optional) changes the operation to find all subshares without exporting them. The found subshares appear in an output report.

This command creates a report to show the progress of the operation. The CLI shows the name of the report after you issue the command. Use show reports *report-name* to view the report.

For example, the following command sequence shares all filer subshares behind the “medarcv~/rcrds” volume through the “ac1.medarch.org” service:

```
bstnA6k(gbl)# end

bstnA6k# cifs export-subshares medarcv /rcrds ac1.medarch.org


% WARNING: Not all nested back-end shares were successfully
exported due to collisions with existing exports. See report
'cifsExportSubshares_20061221140808.rpt' for details.


bstnA6k# ...
```

The warning indicates that some subshares were previously exported. In this example, the warning is expected; an earlier command sequence already exported the "Y2005" share. The report confirms this:

```
bstnA6k# show reports cifsExportSubshares_20061221140808.rpt
**** Cifs Export Subshares Report: Started at Thu Dec 21 14:08:08 2006 ****
**** Software Version: 2.05.000.09902 (Dec 20 2006 21:42:38) [nbuilds]
**** Hardware Platform: ARX-6000

Namespace:     medarcv
Volume:        /rcrds
CIFS Service:  ac1.medarch.org

The following nested back-end shares were successfully exported:

  Back-end Share: CELEBS$
  Exported As:    CELEBS$
  Virtual Path:   /rcrds/VIP_wing

  Back-end Share: Y2004
  Exported As:    Y2004
  Virtual Path:   /rcrds/2004

  Back-end Share: Y2006
  Exported As:    Y2006
  Virtual Path:   /rcrds/2006


Details of failed exports:

  The following exports already exist in this CIFS service:

  Export Name                     Virtual Path
  ------------------------------------------------------------------------------
  Y2005                           /rcrds/2005

**** Total processed:               4
```

```
**** Elapsed time:           00:00:00
**** Cifs Export Subshares Report: DONE at Thu Dec 21 14:08:08 2006 ****
bstnA6k# ...
```

Exposing All Hidden Subshares

For filers with hidden CIFS subshares (such as "CELEBS$" from the above example), you can expose them all as shares from the front-end CIFS service. The front-end shares are renamed without the dollar sign ($) suffix ("CELEBS"). All remaining subshares are also exported, as shown above, without any name changes.

To expose all hidden subshares, use the expose-hidden keyword in the cifs export-subshares command:

> **cifs export-subshares *namespace volume fqdn* expose-hidden [tentative]**
>
> where **expose-hidden** causes the operation to expose all hidden subshares.

For example, the following command sequence exports all subshares from the "ns3~/vol1" volume and exposes any subshares that are hidden:

```
bstnA6k(gbl)# end
bstnA6k# cifs export-subshares ns3 /vol1 ns3.wwmed.org expose-hidden

% INFO: All nested back-end shares were successfully exported. See report
'cifsExportSubshares_200701071217.rpt' for details.

bstnA6k# ...
```

Adding New Subshares

This section only applies to managed volumes. Skip to the next section if you are sharing a direct volume.

The previous sections explained how to export pre-existing subshares, created on the back-end filers before their CIFS shares were imported. To add new subshares, you must directly connect to one of the back-end filers and create them there; a volume manages files and directories, but does not manage share definitions or ACLs.

Note

This is contrary to standard best practices; direct access to the shares behind a managed volume can easily corrupt the volume's metadata. This exception is for new CIFS subshares only.

Once the subshares and their ACLs are created on one filer, you can use the priv-exec sync shares command to find them, duplicate them on all other filers, and make the managed volume aware of them. Then you can use the commands described above to export the new subshares from the front-end CIFS service. The sync utility is described in *CLI Maintenance Guide*; for details on the sync shares command, see "Adding and Synchronizing Filer Subshares (CIFS)" on page 5-30.

Stopping a CIFS Share

Use the no form of the export command to stop CIFS access to a namespace volume:

no export *namespace volume* [as *share-name*]

where

namespace (1-30 characters) is the namespace containing the CIFS share,

volume (1-1024 characters) is the name of a current CIFS share (for example, "alpha"), and

as *share-name* (optional, 1-1024 characters) stops access via the specified share name. If the volume is exported under more than one share name, the share is still supported under the remaining names.

For example, the following command sequence stops the CIFS server at ac1.medarch.org from sharing the "/cifstest" volume in the "medarcv" namespace:

```
bstnA6k(gbl)# cifs ac1.medarch.org
bstnA6k(gbl-cifs[ac1.medarch.org])# no export medarcv /cifstest
bstnA6k(gbl-cifs[ac1.medarch.org])# ...
```

Allowing Clients to Use Windows Management (MMC)

As an alternative to managing the CIFS service from the CLI, Windows clients can use Windows-management applications to manage the service from a remote PC. You can use this feature instead of the export command, above. Authorized clients can use the Microsoft Management Console (MMC) and similar applications to export volumes, list open files, close client sessions, and so on.

From gbl-cifs mode, use the browsing command to enable Windows-management browsing for the namespace behind this service:

browsing *namespace*

where ***namespace*** (1-30 characters) is the namespace that clients can access through Windows-management applications like MMC. A CIFS service can only export volumes from a single namespace, so if you have used the export command already, this must be the same namespace that you previously used.

Only authorized Windows clients can manage the CIFS service. You can configure a group of CIFS clients with Windows-management access (recall "Authorizing Windows-Management (MMC) Access" on page 3-28) and associate it with this service's namespace (also recall "Opening Windows-Management Access (optional, MMC)" on page 7-18). These clients can use MMC and other remote-management applications to fully manage the namespace.

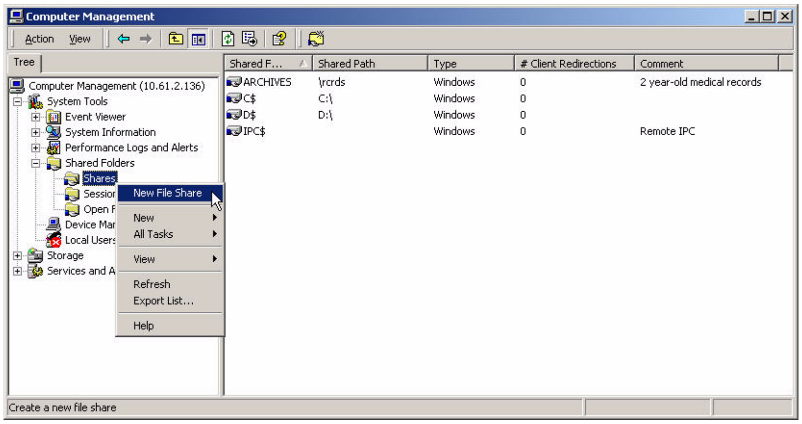
For example, the following command sequence allows authorized clients of the "ac1.medarch.org" service to use Windows management:

```
bstnA6k(gbl)# cifs ac1.medarch.org
bstnA6k(gbl-cifs[ac1.medarch.org])# browsing medarcv
bstnA6k(gbl-cifs[ac1.medarch.org])# ...
```

Client Experience: Using MMC to Manage a Namespace

A properly-enabled client can manage this CIFS service using MMC. For example, the following client session adds a share to the “ac1.medarch.org” service from a Windows 2000 machine. The session starts from Start -> Control Panel -> Administrative Tools -> Computer Management, where you connect to the VIP for the service. This shows a virtual network drive, “C$,” that serves as a container for all of the managed volumes in the namespace. It also shows a separate drive for each direct volume, “D$,” “E$,” and so on. Any of the CIFS service’s shares (exported through the CLI or GUI) also appear here.

From the MMC snap-in interface, you select System Tools -> Shared Folders -> Shares, then pull down a menu option to add a new share:

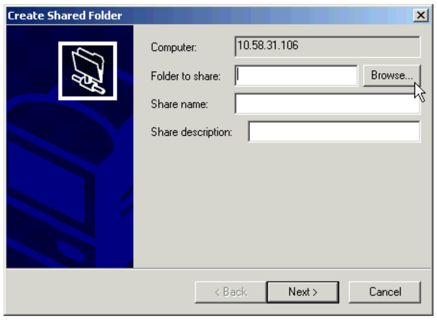

Computer Management

Computer Management (10.61.2.136)

Shared F...	Shared Path	Type	# Client Redirections	Comment
ARCHIVES	\rcrds	Windows	0	2 year-old medical records
C$	C:\	Windows	0	
D$	D:\	Windows	0	
IPC$		Windows	0	Remote IPC

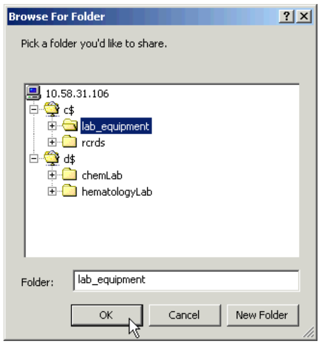
Create a new file share

A pop-up appears. You opt to browse the CIFS service for folders to share:

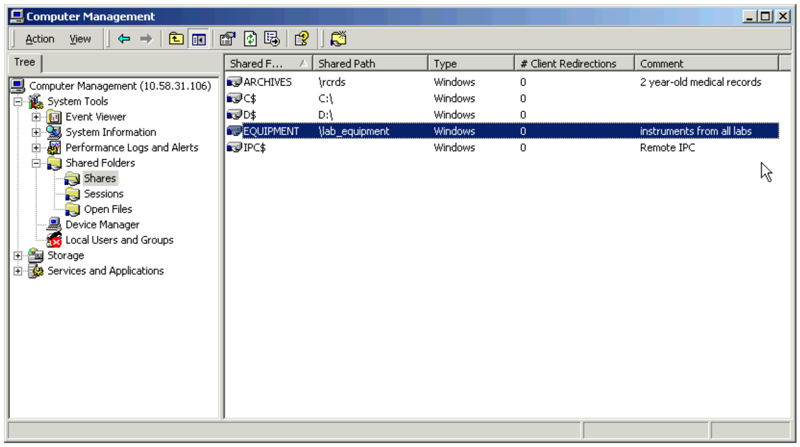

Create Shared Folder

Computer: 10.58.31.106
Folder to share:
Share name:
Share description:

This shows all managed volumes in the CIFS service's namespace under the C drive. Each direct volume in the namespace appears as another drive. In this example, the two managed volumes appear as folders under the C drive, one direct volume appears as the D drive:



You use the interface to export the other managed volume with the share name, "EQUIPMENT." The result is visible from MMC:



Shared F...	Shared Path	Type	# Client Redirections	Comment
ARCHIVES	\rcrds	Windows	0	2 year-old medical records
C$	C:\	Windows	0	
D$	D:\	Windows	0	
EQUIPMENT	\lab_equipment	Windows	0	instruments from all labs
IPC$		Windows	0	Remote IPC

Disallowing Windows-Management Access

Use no browsing to disable Windows management for the current CIFS service:

```
no browsing
```

For example, the following command sequence prevents Windows management of the CIFS service, "beta_service:"

```
bstnA6k(gbl)# cifs beta_service
bstnA6k(gbl-cifs[beta_service])# no browsing
bstnA6k(gbl-cifs[beta_service])# ...
```

Setting a Server Description (optional)

You can optionally set the CIFS-service description that will appear in Windows network browsers. The description appears in the “Remarks” column when you issue a Windows net view command:

```
U:\>net view
Server Name                Remark
-------------------------------------------------------------------------------
\\MEDSERVER
\\ARCHIVE1             A full tree of 5-year-old records
```

Use the description command to set a description for this CIFS service:

description *description*

where ***description*** (1-49 characters) is a quoted string to describe this CIFS service.

For example, the following command sequence sets the CIFS-service description for the global server at “ac1.medarch.org:”

```
bstnA6k(gbl)# cifs ac1.medarch.org
bstnA6k(gbl-cifs[ac1.medarch.org])# description “medical histories and records”
bstnA6k(gbl-cifs[ac1.medarch.org])# ...
```

Removing the Description

Use the no form of the description command to remove the description for this CIFS service:

no description

For example, the following command sequence removes the default description for the global server at “ac1.medarch.org:”

```
bstnA6k(gbl)# cifs ac1.medarch.org
bstnA6k(gbl-cifs[ac1.medarch.org])# no description
```

```
bstnA6k(gbl-cifs[ac1.medarch.org])# ...
```

Enabling CIFS Service

The next step in CIFS configuration is to enable it. Use the enable command from gbl-cifs mode to activate the CIFS service:

enable

For example, the following command sequence enables CIFS for the global server at "ac1.medarch.org:"

```
bstnA6k(gbl)# cifs ac1.medarch.org
bstnA6k(gbl-cifs[ac1.medarch.org])# enable
bstnA6k(gbl-cifs[ac1.medarch.org])# ...
```

Disabling CIFS

Disabling CIFS shuts down all CIFS connections to the global server. From gbl-cifs mode, use no enable to disable CIFS.

no enable

For example, the following command sequence disables CIFS for the "ac1.medarch.org" global server:

```
bstnA6k(gbl)# cifs ac1.medarch.org
bstnA6k(gbl-cifs[ac1.medarch.org])# no enable
bstnA6k(gbl-cifs[ac1.medarch.org])# ...
```

Configuring Kerberos (optional)

For Windows networks that use Kerberos authentication, the CIFS service can provide proxy authentication for its clients. You can only use Kerberos if the namespace behind the CIFS service is configured for it; see "Using Kerberos for Client Authentication" on page 7-15. Additionally, the service's FQDN must belong to a domain in the pre-configured Active-Directory forest (refer back to "Adding an Active-Directory Forest (Kerberos)" on page 3-10).

To enable Kerberos authentication by the CIFS service, you must join the CIFS service to the Active-Directory (AD) domain. This process is similar to adding client computers to the AD domain: this action causes the DC to declare the CIFS service as *Trusted for Delegation*. The CIFS service uses this authority to access back-end filers on behalf of its clients.

Note

Trusting an Acopia server for delegation poses no security threat to your network. Kerberos authentication was designed with delegation in mind to provide a clean way of carrying identity through *n*-tiered application systems. For more information, refer to IETF RFC 1510 or the Microsoft white paper on Kerberos authentication (http://www.microsoft.com/windows2000/techinfo/howitworks/security/kerberos.asp).

Use the gbl-cifs domain-join command as follows:

domain-join *domain-name* [ou "*organizational-unit*"]

where

domain-name (1-256 characters) identifies the DC domain. This domain must be defined in the AD forest; see "Adding an Active-Directory Forest (Kerberos)" on page 3-10.

organizational-unit (optional, 1-512 characters) is the organizational unit (OU) to join. An *OU* is a group of similar accounts or machines that is managed by a particular administrator. The default is "Computers" at the root of the domain. If the OU does not exist, the domain-join operation fails.

The CLI prompts for the username and password. Enter a username with the following permissions in the domain:

- "Add workstations to domain" and
- "Enable computer and user accounts to be trusted for delegation."

The user account must have the following additional permissions in the OU:

- "Create computer objects" and
- "Delete computer objects."

For example, the following command sequence enables the CIFS service at "ac1.medarch.org," then joins the domain under username "acoadmin:"

```
bstnA6k(gbl)# cifs ac1.medarch.org
bstnA6k(gbl-cifs[ac1.medarch.org])# enable
bstnA6k(gbl-cifs[ac1.medarch.org])# domain-join MEDARCH.ORG
Username: acoadmin
Password: aapasswd

  'ac1' successfully joined the domain.

bstnA6k(gbl-cifs[ac1.medarch.org])# ...
```

Support for Both NTLM and Kerberos

The domain-join operation does not preclude any clients from authenticating with NTLM; the CIFS service can support both authentication protocols concurrently. You can use this feature for a network transition from NTLM to Kerberos. (NTLM authentication is configured in the namespace; recall "Identifying the NTLM Authentication Server" on page 7-16).

Each client negotiates the authentication protocol with the CIFS service when it initiates a CIFS session. If both protocols are configured and functional, the CIFS service indicates its preference for Kerberos but accepts NTLM from its clients. The CIFS service uses the chosen protocol to authenticate the client, then uses the same protocol to interact with all back-end filers. The switch does not translate between NTLM and Kerberos.

For filer interaction that does not directly involve any clients, such as file migrations, the ARX uses NTLM only.

Using Dynamic DNS (Kerberos)

Every front-end CIFS service that uses Kerberos must be registered with the network's Domain Name Service (DNS), which maps IP addresses to FQDNs. In each AD domain, one or more *name servers* take these DNS registrations; often, DNS runs on the DC itself. You can manually update the local name server with the hostname-to-IP mapping for the CIFS service, or you can configure the service to use *dynamic DNS*. With dynamic DNS, the CIFS service automatically registers its hostname-to-IP mapping, and updates it whenever a failover or configuration change makes an update necessary.

RFCs 1034 and 1035 define basic DNS, and RFC 3645 defines Microsoft-specific authentication extensions to for dynamic DNS. The ARX implementation of dynamic DNS adheres to all of these RFCs.

Before you use dynamic DNS, the name server(s) for this service's Windows Domain must be included in the AD forest. For instructions on setting up name servers for the AD forest, recall "Identifying a Dynamic-DNS Server" on page 3-12. The Windows Domain for this CIFS service is determined by the service's global server; use show global server to see the domain (recall "Showing One Global-Server" on page 10-13).

Use the dynamic-dns command to choose a name for this service and register it with a name server. This sends an "A" (Address) record that maps the virtual server's VIP to a host name:

dynamic-dns *host-name*

where ***host-name*** (1-255 characters) is a host name you choose for this CIFS service. This is mapped to the virtual server's VIP in the "A" record. If you enter a host name without any domain (for example, "myservice"), the global server's Windows domain is appended to it. If you specify a domain (for example, "myservice.myco.net"), the domain must match the Windows domain for the global server.

You can repeat the command to enter additional aliases for this CIFS service. All of these host names map to the CIFS service's VIP; Windows clients can use any of them to access the CIFS service.

For example, this command sequence registers two host names, "test" and "ac1," for the current CIFS service:

```
bstnA6k(gbl)# cifs ac1.medarch.org
bstnA6k(gbl-cifs[ac1.medarch.org])# dynamic-dns test
bstnA6k(gbl-cifs[ac1.medarch.org])# dynamic-dns ac1
bstnA6k(gbl-cifs[ac1.medarch.org])# ...
```

Replacing Back-End Services

Through DNS aliasing, you can transparently replace the CIFS services on existing back-end filers. For example, suppose you had the following CIFS filers and shares before the ARX was installed:

- \\fs1.medarch.org\xrays

- \\fs2.medarch.org\lab_data
- \\fs7.medarch.org\tests

You can import each of these shares into a single namespace, where each share is in a separate volume. From gbl-cifs mode, you can then export each CIFS volume under its original share name ("xrays," "lab_data," and "tests," respectively). Finally, you can use the dynamic-dns command to register all three of the original host names as DNS aliases for the CIFS service ("fs1," "fs2," and "fs5"). Clients can then access all three shares using the same host names and share names.

Note

Clients cannot yet use Kerberos authentication with these DNS aliases; the AD domain still needs a mapping of these *service principal names* (SPNs) to the actual FQDN for the CIFS service. A later section describes a method for supporting aliases with Kerberos.

No further action is required for the CIFS-service FQDN itself. Windows clients can connect to the service's FQDN and authenticate with Kerberos without any additional configuration.

Update Schedule for DNS "A" Records

The CIFS service updates the DNS database as soon as you enter the dynamic-dns command. It re-registers the "A" record once per day at 1:00 AM (local time), whenever the ARX reboots or fails over, whenever the VIP for this service changes, or if ever the VIP or CIFS service is deleted.

To update all DNS records immediately, you can exit to priv-exec mode and use the dynamic-dns update command:

dynamic-dns update [*fqdn*]

where ***fqdn*** (optional: 1-128 characters) narrows the scope of the update to one CIFS service. This is the FQDN that identifies the CIFS service and its global server. If you omit this, the ARX updates all host names for all CIFS services; this can be network-intensive.

The CLI prompts for confirmation if you choose to update for all CIFS services. Enter **yes** to continue.

For example, the following commands exit to priv-exec mode and update all of the DNS entries for a particular CIFS service, "ac1.medarch.org:"

```
bstnA6k(gbl-cifs[ac1.medarch.org])# end

bstnA6k# dynamic-dns update ac1.medarch.org

bstnA6k# ...
```

Removing a Host Name

You can use the no dynamic-dns command to remove one host name from the current CIFS service. This causes the CIFS service to withdraw all references to this host name from DNS. If you remove all registered host names for this CIFS service, you must manually update the domain's DNS server to support Kerberos.

no dynamic-dns *host-name*

where ***host-name*** (1-255 characters) is the host name to remove.

For example, this command sequence removes "test" as a DNS alias for the current CIFS service:

```
bstnA6k(gbl)# cifs ac1.medarch.org

bstnA6k(gbl-cifs[ac1.medarch.org])# no dynamic-dns test

bstnA6k(gbl-cifs[ac1.medarch.org])# ...
```

Showing Dynamic-DNS Status

The show dynamic-dns command displays the status of all host-name registrations:

show dynamic-dns [*fqdn*]

where ***fqdn*** (optional: 1-128 characters) is the fully-qualified domain name (for example, www.company.com) for the CIFS service's global server. If you omit this, the output includes all CIFS services.

You can invoke this command from any mode.

For each CIFS service, this lists all host names, whether or not their "A" records were properly registered, the time of the last registration, and the IP address of the DNS server that accepted the record. For example, this lists the single host name for the "ac1.medarch.org" service. The host name and VIP, "ac1.MEDARCH.ORG/192.168.25.15," was successfully registered at the local DNS server:

```
bstnA6k> show dynamic-dns ac1.medarch.org
```

```
Svc     Global Server                   Domain Name
-------------------------------------------------------------------------------
CIFS    ac1.MEDARCH.ORG                 MEDARCH.ORG

  Status  Host Name                                           VIP
          Operation      Retries   Last Update                DNS Server
  -----------------------------------------------------------------------------
  OK      ac1.MEDARCH.ORG                                     192.168.25.15
          Add                0     Wed Oct  4 06:56:24 2006   192.168.25.104

bstnA6k>
```

Clearing Records of Failed Deletes

The CIFS service performs two dynamic-DNS operations: *add* and *remove*. The dynamic-dns command triggers an add operation, and a remove occurs after no dynamic-dns. If an add operation fails, the service retries once per minute indefinitely. If a remove operation fails, the service stops retrying after the 15th attempt. After the 15th failure, the remove operation gets a "Failed" status in the show dynamic-dns output; you will have to remove the "A" record at the name server itself. For your convenience, these records remain in the show dynamic-dns output until someone clears them.

From priv-exec mode, you can use the clear dynamic-dns command to clear all failed deletes from the show output:

clear dynamic-dns

The CLI prompts for confirmation before deleting all the records on the system; enter **yes** to proceed.

For example, the following command sequence shows a failed remove operation (highlighted in bold text), clears it, and shows that it is cleared:

```
bstnA6k# show dynamic-dns

Svc     Global Server                   Domain Name
-------------------------------------------------------------------------------
CIFS    ac1.MEDARCH.ORG                 MEDARCH.ORG
```

```
  Status  Host Name                                           VIP
          Operation      Retries   Last Update                DNS Server
  ---------------------------------------------------------------------------
  Failed  test.MEDARCH.ORG                                    192.168.25.15
          Remove            15     Wed Oct  4 07:09:33 2006   192.168.25.102

  OK      ac1.MEDARCH.ORG                                     192.168.25.15
          Add                0     Wed Oct  4 06:56:24 2006   192.168.25.104

  Retry   fs7.MEDARCH.ORG                                     192.168.25.15
          Add               19     Wed Oct  4 07:12:23 2006   192.168.25.104

bstnA6k# clear dynamic-dns
Clear failed dynamic DNS entries? [yes/no] yes
bstnA6k# show dynamic-dns

Svc     Global Server                   Domain Name
-----------------------------------------------------------------------------
CIFS    ac1.MEDARCH.ORG                 MEDARCH.ORG

  Status  Host Name                                           VIP
          Operation      Retries   Last Update                DNS Server
  ---------------------------------------------------------------------------
  OK      ac1.MEDARCH.ORG                                     192.168.25.15
          Add                0     Wed Oct  4 06:56:24 2006   192.168.25.104

  Retry   fs7.MEDARCH.ORG                                     192.168.25.15
          Add               19     Wed Oct  4 07:12:23 2006   192.168.25.104

bstnA6k# ...
```

Supporting Aliases with Kerberos

This section does not apply to a CIFS service that only uses NTLM authentication. You can also skip this section if you have not registered a WINS name, any WINS aliases, or any DNS aliases for your CIFS service.

When a CIFS service joins its AD domain, it registers its FQDN name in the Active-Directory database. The FQDN for the CIFS service becomes the *service principal name* (SPN) for the service: the DCs uses this name to identify the CIFS service for Kerberos authentications. If a client uses an alias name to connect to a CIFS service, a DC cannot use WINS or DNS to translate the alias to the actual SPN. This causes Kerberos authentication to fail.

You may have already registered CIFS-service aliases through any of three commands described earlier. From gbl-gs-vs mode, you can use wins-name to register a different name with the local WINS server (recall "Setting the NetBIOS Name (optional, CIFS)" on page 10-7) and you can use wins-alias to register one or more WINS aliases ("Adding a NetBIOS Alias" on page 10-8). From gbl-cifs mode, you can also register DNS aliases through dynamic DNS, as described above ("Using Dynamic DNS (Kerberos)" on page 11-28).

If you set any WINS or DNS alias, you must map the alias to the CIFS service's SPN. With this mapping, Kerberos clients that connect to the alias can successfully authenticate at any DC in the domain. You can use a free command-line utility from Microsoft to set up the mapping: setspn.exe. Follow this URL and search the site for "setspn.exe download":

http://www.microsoft.com/

Install setspn.exe on a Windows machine in the AD domain, open a DOS shell, and go to the directory where it is installed (C:\Program Files\Resource Kit\, by default). Use the following syntax in the DOS shell to map an alias to a CIFS-service name:

setspn -A HOST/*alias cifs-service*

where

alias is the WINS alias or DNS alias that you registered earlier, and

cifs-service is the FQDN that identifies the CIFS front-end service.

For each alias, we recommend mapping both the simple host name and the full FQDN. For example, the following DOS-command sequence maps three DNS aliases to the “ac1.medarch.org” CIFS service:

```
C:\Program Files\Resource Kit> setspn -A HOST/fs1 ac1.medarch.org
C:\Program Files\Resource Kit> setspn -A HOST/fs1.medarch.org ac1.medarch.org
C:\Program Files\Resource Kit> setspn -A HOST/fs2 ac1.medarch.org
C:\Program Files\Resource Kit> setspn -A HOST/fs2.medarch.org ac1.medarch.org
C:\Program Files\Resource Kit> setspn -A HOST/fs7 ac1.medarch.org
C:\Program Files\Resource Kit> setspn -A HOST/fs7.medarch.org ac1.medarch.org
```

With these SPN mappings, Windows clients can connect to the CIFS service through any of these aliases using Kerberos authentication.

Listing CIFS Services

Use the show cifs-service command to list all global servers that offer CIFS services:

show cifs-service

For example:

```
bstnA6k> show cifs-service

Domain Name                 Namespace         Description
-------------------------------------------------------------------------------
ac1.medarch.org             medarcv           medical histories and records
insur.medarch.org           insur             insurance-claim records

bstnA6k>
```

Focusing on One CIFS Service

To focus on a particular CIFS front-end service, specify the FQDN for the service in the show cifs-service command:

show cifs-service *fqdn*

where ***fqdn*** (1-128 characters) is the fully-qualified domain name (for example, www.company.com) for the global server.

This shows a configuration summary followed by a table of CIFS shares.

For example:

```
bstnA6k> show cifs-service ac1.medarch.org

Domain Name:    ac1.medarch.org
Description:    medical histories and records
Namespace:      medarcv
State:          Enabled

  Share Name                 Directory                                State
  ----------------------------------------------------------------------------
  ARCHIVES                   /rcrds                                   Online
  Y2005                    S /rcrds/2005                              Online
  CELEBS                   S /rcrds/VIP_wing                          Online
  Y2004                    S /rcrds/2004                              Online
  CELEBS$                  S /rcrds/VIP_wing                          Online
  Y2006                    S /rcrds/2006                              Online

  S = filer-subshare export

bstnA6k>
```

Showing Details for a CIFS Service

You can add an optional argument, detailed, for details about each CIFS share in the service:

show cifs-service *fqdn* detailed

For example:

```
bstnA6k> show cifs-service ac1.medarch.org detailed

Domain Name:    ac1.medarch.org
Description:    medical histories and records
Namespace:      medarcv
State:          Enabled
```

```
Shares
------
  ARCHIVES
    Directory        /rcrds
    Description      2 year-old medical records
    State            Online
    Filer-subshare   No

  Y2005
    Directory        /rcrds/2005
    Description
    State            Online
    Filer-subshare   Yes

  CELEBS
    Directory        /rcrds/VIP_wing
    Description
    State            Online
    Filer-subshare   Yes (hidden)

  Y2004
    Directory        /rcrds/2004
    Description
    State            Online
    Filer-subshare   Yes

  CELEBS$
    Directory        /rcrds/VIP_wing
    Description
    State            Online
    Filer-subshare   Yes

  Y2006
    Directory        /rcrds/2006
```

```
      Description
      State            Online
      Filer-subshare   Yes

bstnA6k>
```

Showing All CIFS Services

To show all CIFS front-end services, use show cifs-service all:

show cifs-service all [detailed]

where **detailed** (optional) adds details to the CIFS shares.

For example, this shows a summary view of every CIFS service on the ARX:

```
bstnA6k> show cifs-service all

Domain Name:    ac1.medarch.org
Description:    medical histories and records
Namespace:      medarcv
State:          Enabled

  Share Name                Directory                                State
  ---------------------------------------------------------------------------
  ARCHIVES                  /rcrds                                   Degraded
  Y2005                   S /rcrds/2005                              Degraded
  CELEBS$                 S /rcrds/VIP_wing                          Degraded
  Y2004                   S /rcrds/2004                              Degraded
  Y2006                   S /rcrds/2006                              Degraded

  S = filer-subshare export


Domain Name:    insur.medarch.org
Description:    insurance-claim records
Namespace:      insur
State:          Enabled
```

```
  Share Name                Directory                               State
  ---------------------------------------------------------------------------
  CLAIMS                    /claims                                 Online
...
```

Sample - Configuring a CIFS Front-End Service

The following command sequence sets up CIFS service on a global server called “ac1.medarch.org:”

```
bstnA6k(gbl)# cifs ac1.medarch.org
bstnA6k(gbl-cifs[ac1.medarch.org])# show global-config namespace medarcv
;=============================== namespace ================================
namespace medarcv
  kerberos-auth
  ntlm-auth-server dc1
...

  volume /rcrds
    filer-subshares replicate
    modify
...
bstnA6k(gbl-cifs[ac1.medarch.org])# export medarcv /rcrds as ARCHIVES
description "2-year-old medical records"
bstnA6k(gbl-cifs[ac1.medarch.org])# description "medical histories and records"
bstnA6k(gbl-cifs[ac1.medarch.org])# enable
bstnA6k(gbl-cifs[ac1.medarch.org])# domain-join MEDARCH.ORG
Username: acoadmin
Password: aapasswd

  'ac1' successfully joined the domain.

bstnA6k(gbl-cifs[ac1.medarch.org])# exit
bstnA6k(gbl)# ...
```

Removing a CIFS Service

You can remove a CIFS service from a global server to both disable the service and remove its configuration.Use the no form of the cifs command to remove an CIFS-service configuration from a global server:

no cifs *fqdn*

where ***fqdn*** is the fully-qualified domain name (for example, “www.organization.org”) for the global server.

The CLI prompts for confirmation before removing the service; enter **yes** to proceed.

For example, the following command sequence removes the CIFS-service offering for ac1.medarch.org:

```
bstnA6k(gbl)# no cifs ac1.medarch.org
Delete CIFS service on 'ac1.medarch.org'? [yes/no] yes
bstnA6k(gbl)#
```

Removing All of a Volume's Front-End Exports

You can use a single command to remove all of the front-end exports, NFS and/or CIFS, for a given volume. This is convenient for a volume that has been exported through multiple global servers and front-end services. The remove namespace command has been described in previous chapters for removing a volume (see "Removing a Direct Volume" on page 8-37 or "Removing a Managed Volume" on page 9-71) or an entire namespace ("Removing a Namespace" on page 7-27); add the optional exports-only keyword to remove only the front-end exports:

remove namespace *name* volume *volume* exports-only [timeout *seconds*] [sync]

where:

name (1-30 characters) is the name of the namespace,

volume (optional, 1-1024 characters) is the path name of the volume,

exports-only is the option to remove only the front-end exports for the volume,

seconds (optional, 300-10,000) sets a time limit on each of the removal's component operations, and

sync (optional) waits for the removal to finish before returning. With this option, the CLI lists the configuration objects as it removes them.

The CLI prompts for confirmation before removing the exports. Enter **yes** to continue. This operation generates a report named "removeNs_*namespace*_*date*.rpt" if you omit the sync option.

For example, this command sequence exits to priv-exec mode and then synchronously removes all front-end exports from the "insur_bkup~/insurShdw" volume:

```
prtlndA1k(gbl)# end
prtlndA1k# remove namespace insur_bkup volume /insurShdw exports-only sync

Remove exports for namespace 'insur_bkup', volume '/insurShdw'? [yes/no] yes
% INFO: Removing service configuration for namespace insur_bkup
% INFO: Removing NFS services for namespace insur_bkup
% INFO: Removing CIFS services for namespace insur_bkup
% INFO: no export insur_bkup /insurShdw as CLAIMS_BKUP
prtlndA1k# ...
```

Showing All Front-End Services

Front-end services are identified by the FQDN of their respective global servers. Use the show global service command to show all front-end services configured on the ARX:

show global service

For example:

```
bstnA6k(gbl)# show global service

Domain Name                   Service     State
--------------------------------------------------
www.wwmed.com                 NFS         Enabled
acopiaFiler                   NFS         Enabled
ac1.medarch.org               CIFS        Enabled
insur.medarch.org             CIFS        Enabled
insur.medarch.org             NFS         Enabled
bstnA6k(gbl)# ...
```

Showing Front-End Services for One Global-Server

To see the front-end services for one global server, identify a particular global server with the show global service command:

show global service *fqdn*

where ***fqdn*** (1-128 characters) is the fully-qualified domain name (for example, www.company.com) for the global server.

For example, the following command shows the front-end services for the global server at “www.wwmed.com:”

```
bstnA6k(gbl)# show global service www.wwmed.com
```

```
Domain Name                  Service     State
--------------------------------------------------
www.wwmed.com                NFS         Enabled
bstnA6k(gbl)# ...
```

Showing Front-End Services per Virtual-Server

You can show the front-end services running at each virtual server, with the VIP and current health of each service. To see the front-end services grouped by their virtual servers, use the show virtual service command:

show virtual service

For example:

```
bstnA6k(gbl)# show virtual service
Switch   bstnA6k
------------------------
  Global Server           Virtual IP Address      Service     State
  -------------------------------------------------------------------------
  acopiaFiler             192.168.25.12           NFS         Ready
  www.wwmed.com           192.168.25.10           NFS         Ready
  ac1.medarch.org         192.168.25.15           CIFS        Ready
  insur.medarch.org       192.168.25.14           CIFS        Ready
  insur.medarch.org       192.168.25.14           NFS         Ready


bstnA6k(gbl)# ...
```

In a redundant pair, this shows both peers and the virtual services that are running on each. For example:

```
prtlndA1k# show virtual service
Switch   prtlndA1k
------------------------
  Global Server           Virtual IP Address      Service     State
  -------------------------------------------------------------------------
  www.nemed.com           192.168.74.91           NFS         Ready
  www.insurBkup.com       192.168.74.92           CIFS        Ready

Switch   prtlndA1kB
------------------------
  Global Server           Virtual IP Address      Service     State
  -------------------------------------------------------------------------


prtlndA1k# ...
```

Showing the Services at the Redundant Peer

To focus on one peer in the redundant pair, identify the peer switch at the end of the command:

show virtual service *peer-name*

where ***peer-name*** (1-128 characters) identifies the peer by its hostname.

For example, this shows the services running on “prtlndA1k” from its redundant peer, “prtlndA1kB:”

```
prtlndA1kB# show virtual service prtlndA1k
Switch   prtlndA1k
------------------------
  Global Server           Virtual IP Address      Service     State
  -------------------------------------------------------------------------
  www.nemed.com           192.168.74.91           NFS         Ready
  www.insurBkup.com       192.168.74.92           CIFS        Ready

Switch   prtlndA1kB
------------------------
  Global Server           Virtual IP Address      Service     State
  -------------------------------------------------------------------------


prtlndA1kB# ...
```

Showing Server Maps

You can show the map between front-end services and the back-end servers behind them. From any mode, use the show server-mapping command:

show server-mapping

This displays a two-column table, where the left column shows the client-side view and the right column shows the server side. Each front-end export has its own listing of back-end filers. For example, the following command shows all front-end exports on the “bstnA6k” switch with their backing filers:

```
bstnA6k(gbl)# show server-mapping

Virtual Server                Namespace/Volume
   Virtual Path                  Physical Server
----------------------------------------------------------------------
192.168.25.10:/acct           wwmed:/acct

                                  das1:/exports/budget
                                  das3:/data/acct2
                                  das7:/lhome/it5
                                  das8:/work1/accting
                                  nas1:/vol/vol1/meta1*


192.168.25.12:/vol            medco:/vol

   vol0/corp                      nas1:/vol/vol0/direct/shr
   vol0/notes                     nas1:/vol/vol0/direct/notes
   vol1/mtgMinutes                nas2:/vol/vol1/direct/mtgs
   vol1/sales                     nas2:/vol/vol1/direct/export
   vol2                           nas3:/vol/vol2/direct/data


192.168.25.14:/claims         insur:/claims

                                  nas1:/vol/vol1/meta2*
                                  nas1:/vol/vol1/NTFS-QTREE/insurance
                                  nasE1:/root_vdm_4/patient_records
```

```
\\192.168.25.14\CLAIMS       insur:/claims

                                 nas1:/vol/vol1/meta2*
                                 \\nas1\insurance
                                 \\nasE1\patient_records


\\192.168.25.14\SPECS        insur:/claims

                                 nas1:/vol/vol1/meta2*
                                 \\nas1\insurance
                                 \\nasE1\patient_records


\\192.168.25.14\STATS        insur:/claims

                                 nas1:/vol/vol1/meta2*
                                 \\nas1\insurance
                                 \\nasE1\patient_records


\\192.168.25.15\ARCHIVES     medarcv:/rcrds

                                 \\fs1\histories
                                 \\fs2\bulkstorage
                                 \\fs4\prescriptions
                                 nas1:/vol/vol1/meta3*


\\192.168.25.15\CELEBS       medarcv:/rcrds

                                 \\fs1\histories
                                 \\fs2\bulkstorage
                                 \\fs4\prescriptions
                                 nas1:/vol/vol1/meta3*


\\192.168.25.15\Y2004        medarcv:/rcrds

                                 \\fs1\histories
                                 \\fs2\bulkstorage
```

```
                                    \\fs4\prescriptions
                                    nas1:/vol/vol1/meta3*


\\192.168.25.15\Y2005           medarcv:/rcrds

                                    \\fs1\histories
                                    \\fs2\bulkstorage
                                    \\fs4\prescriptions
                                    nas1:/vol/vol1/meta3*



Where * denotes metadata only physical server.

bstnA6k(gbl)# ...
```

With Filer IP Addresses

You can add the ip-addresses option to the end of the command to show the IP addresses of the back-end filers, rather than their external-filer names:

show server-mapping ip-addresses

For example this command shows the same information as above, but with IP addresses in place of the external-filer names:

```
bstnA6k(gbl)# show server-mapping ip-addresses

Virtual Server                  Namespace/Volume
   Virtual Path                    Physical Server
-----------------------------------------------------------------------
192.168.25.10:/acct             wwmed:/acct

                                    192.168.25.19:/exports/budget
                                    192.168.25.23:/data/acct2
                                    192.168.25.24:/lhome/it5
                                    192.168.25.25:/work1/accting
                                    192.168.25.21:/vol/vol1/meta1*


192.168.25.12:/vol              medco:/vol
```

```
   vol0/corp                     192.168.25.21:/vol/vol0/direct/shr
   vol0/notes                    192.168.25.21:/vol/vol0/direct/notes
...

bstnA6k(gbl)# ...
```

Showing the Servers Behind One Virtual Server

To focus on one virtual server, add its VIP to the end of the command:

show server-mapping virtual-ip *vip* [ip-addresses]

where

vip identifies the VIP (for example, 172.16.77.75), and

ip-addresses (optional) is explained above.

For example, the following command shows the filers behind “192.168.25.15.” This VIP has multiple exports (described in the next chapter), and every export draws from the same set of filers:

```
bstnA6k(gbl)# show server-mapping virtual-ip 192.168.25.15

Virtual Server                Namespace/Volume
   Virtual Path                  Physical Server
----------------------------------------------------------------------
\\192.168.25.15\ARCHIVES     medarcv:/rcrds

                                  \\fs1\histories
                                  \\fs2\bulkstorage
                                  \\fs4\prescriptions
                                  nas1:/vol/vol1/meta3*


\\192.168.25.15\CELEBS       medarcv:/rcrds

                                  \\fs1\histories
                                  \\fs2\bulkstorage
                                  \\fs4\prescriptions
                                  nas1:/vol/vol1/meta3*
```

```
\\192.168.25.15\Y2004        medarcv:/rcrds

                                 \\fs1\histories
                                 \\fs2\bulkstorage
                                 \\fs4\prescriptions
                                 nas1:/vol/vol1/meta3*


\\192.168.25.15\Y2005        medarcv:/rcrds

                                 \\fs1\histories
                                 \\fs2\bulkstorage
                                 \\fs4\prescriptions
                                 nas1:/vol/vol1/meta3*



Where * denotes metadata only physical server.

bstnA6k(gbl)# ...
```

Showing the Servers Behind One Namespace

You can also show the physical servers behind one namespace:

show server-mapping namespace *name* [ip-addresses]

where

name (1-30 characters) identifies the namespace, and

ip-addresses (optional) was explained earlier.

If multiple virtual servers provide service for the namespace, this command shows all of them.

For example, the following command shows the filers behind the “wwmed” namespace. This shows IP addresses instead of external-filer names:

```
bstnA6k(gbl)# show server-mapping namespace wwmed ip-addresses

Virtual Server                 Namespace/Volume
   Virtual Path                   Physical Server
----------------------------------------------------------------------
192.168.25.10:/acct            wwmed:/acct

                                   192.168.25.19:/exports/budget
                                   192.168.25.23:/data/acct2
                                   192.168.25.24:/lhome/it5
                                   192.168.25.25:/work1/accting
                                   192.168.25.21:/vol/vol1/meta1*



Where * denotes metadata only physical server.
Where ** denotes a direct volume mapped to a namespace.
bstnA6k(gbl)# ...
```

Showing Server Status

The status keyword shows high-level status for each of the servers. The status is shown for each virtual server as well as each of the shares behind it. This provides a high-level view of each virtual server’s health:

show server-mapping status [ip-addresses]

where **ip-addresses** (optional) shows filer IPs instead of their external-filer names, as explained above.

To examine the configuration and status for a each share, use the show namespace command. See “Showing Namespace Details” on page 7-3.

For example, the following command shows that all virtual servers are ready and all shares are online:

```
bstnA6k(gbl)# show server-mapping status

Virtual Server
        Physical Server                                                 Status
---------------------------------------------------------------    --------
192.168.25.12:/vol                                                  Ready

        nas1:/vol/vol0/direct/shr                                   Online
        nas1:/vol/vol0/direct/notes                                 Online
        nas2:/vol/vol1/direct/export                                Online
        nas2:/vol/vol1/direct/mtgs                                  Online
        nas3:/vol/vol2/direct/data                                  Online

192.168.25.10:/acct                                                 Ready

        das1:/exports/budget                                        Online
        das8:/work1/accting                                         Online
        das3:/data/acct2                                            Online
        das7:/lhome/it5                                             Online

\\192.168.25.15\ARCHIVES                                            Ready

        \\fs4\prescriptions                                         Online
        \\fs1\histories                                             Online
        \\fs2\bulkstorage                                           Online

\\192.168.25.15\Y2005                                               Ready

        \\fs4\prescriptions                                         Online
        \\fs1\histories                                             Online
        \\fs2\bulkstorage                                           Online

\\192.168.25.15\CELEBS                                              Ready

        \\fs4\prescriptions                                         Online
        \\fs1\histories                                             Online
        \\fs2\bulkstorage                                           Online

\\192.168.25.15\Y2004                                               Ready

        \\fs4\prescriptions                                         Online
        \\fs1\histories                                              Online
        \\fs2\bulkstorage                                            Online

\\192.168.25.14\CLAIMS                                               Ready

        \\nas1\insurance                                             Online
        \\nasE1\patient_records                                      Online

\\192.168.25.14\SPECS                                                Ready

        \\nas1\insurance                                             Online
        \\nasE1\patient_records                                      Online

\\192.168.25.14\STATS                                                Ready

        \\nas1\insurance                                             Online
        \\nasE1\patient_records                                      Online

192.168.25.14:/claims                                                Ready

        nas1:/vol/vol1/NTFS-QTREE/insurance                          Online
        nasE1:/root_vdm_4/patient_records                            Online


bstnA6k(gbl)# ...
```

Chapter 12

Policy for Balancing Capacity

Namespace *policy* uses file migration and replication to balance the usage of various back-end filers. This chapter explains how to configure policies for managing free space. Use this chapter to

- show policy parameters,
- add share farms (groups of shares) to a namespace,
- configure capacity balancing for the share farm,
- create schedules for rules to follow,
- pause all policy rules in a volume (perhaps on a schedule),
- drain all files off of a share or share farm, and/or
- remove all policy configuration from a namespace or volume.

The final sections discuss the side-effects of file and directory migrations, particularly in multi-protocol (CIFS and NFS) namespaces.

Direct volumes, which contain no metadata, do not support any policy. This chapter is relevant to managed volumes only.

The policies described in this chapter are indiscriminate about the types of files replicated and migrated; they are focused on usage levels at the back end. There are other policies that you can use to select files and migrate them to one or more back-end filers, share farms, or classes of storage. The next chapter describes how to define a fileset to select a group of files for management. The chapter after that explains how to define rules for migrating filesets. Skip ahead to configure file-based policies.

Before You Begin

You must configure a namespace and at least one managed volume before you configure the policies described in this chapter. See Chapter 7, *Configuring a Namespace*, and Chapter 9, *Adding a Managed Volume*.

Concepts and Terminology

A *rule* is a condition or set of conditions for moving files between back-end storage devices. Namespace *policy* is a series of namespace rules.

The ARX can move files between external storage devices. This is called *migration*.

When files migrate from one share to another, their parent directories are duplicated on the destination share. The directories are said to be *striped* across the two shares. The original directory, the one that was first-imported, is called the *master directory*. A new file or directory goes to its parent's master directory by default, so that directory trees tend to grow on a single back-end share.

Showing All Policy Rules

As you configure policy, it can be useful to review the current policy settings. Use the show policy command to get a full list of all policies:

show policy

This shows a list of rules and share farms (explained below), grouped by namespaces.

For example:

```
bstnA6k# show policy


Status
Namespace                Volume                   Rule                     Vol. Scan
Migration
-------------------------  -------------------------  -------------------------
----------------------------------------
wwmed                      /acct                      docs2das8                  Volume
Paused          Volume Paused
```

```
wwmed                  /acct                  fm1                    Complete
Complete
medarcv                /rcrds                 dailyArchive            Complete
Complete
medarcv                /rcrds                 medFm                  Complete
Complete
bstnA6k# ...
```

Showing Details

Add the **details** keyword to the end of the command to show details for all policies on the ARX:

show policy details

The output is divided into namespaces, volumes, and rules. All namespaces and volumes are listed, even those without any rules or share farms. Each rule has one or more tables describing its configuration, its latest status, and various statistics. Share farms appear at the same level as rules. For example:

```
bstnA6k# show policy details

Namespace:          medco
Namespace:          wwmed
Volume:             /acct

  Share Farm:         fm1

                  Placement Frequency   Freespace Status
     Share Name   Count/Total           Free    Size    Pct Free
     -----------  --------------------  ----------------------
     bills        179 / 767       23 %  19G     34G     55 %
     bills2       41  / 767       5  %  4.4G    4.9G    91 %
     budget       547 / 767       71 %  59G     62G     94 %


  New File Placement Rule:  fm1

    Configuration:
      Constrain Files:                           No
      Constrain Directories:                     No
      Balance Mode:                              Capacity
      Maintain Freespace:                        2G
      Auto Migrate:                              2G

      State:                                     Enabled

    Status:
      Volume Scan Status:                        Complete
      File Migration Status:                     Complete
      New File Placement Status:                 Enabled

    Cumulative Statistics:
      Total Files Migrated:                      0
      Total Directories Promoted:                0
      Total Failed Migrations:                   0
      Total Failed Directory Promotes:           0
      Total Retried Migrations:                  0
      Total Canceled Migrations:                 0
      Total Hard Links Skipped:                  0
      Total Files Placed Inline:                 0
      Total File Renames Processed Inline:       0
      Total Directories Placed Inline:           0
      Total Directory Renames Processed Inline:  0
      Number of Scans Performed:                 0

    Queue Statistics:
      First-time Migrates:                       0
      Requeued Migrates:                         0
      Queued Directory Promotes:                 0



  Place Rule:       docs2das8
```

```
Configuration:
  From fileset:                           bulky (files only)
  Target share:                           bills
  Report:                                 docsPlc, Verbose
  Migrate limit:                          50G
  Volume Scan:                            Enabled
  Inline Notifications:                   Enabled
  Promote Directories:                    Disabled

  State:                                  Enabled

Status:
  Volume Scan Status:                     Complete
  File Migration Status:                  Complete
  New File Placement Status:              Enabled

Cumulative Statistics:
  Total Files Migrated:                   68
  Total Directories Promoted:             0
  Total Failed Migrations:                0
  Total Failed Directory Promotes:        0
  Total Retried Migrations:               240
  Total Canceled Migrations:              0
  Total Hard Links Skipped:               0
  Total Files Placed Inline:              39
  Total File Renames Processed Inline:    0
  Total Directories Placed Inline:        0
  Total Directory Renames Processed Inline: 0
  Number of Scans Performed:              1

Queue Statistics:
  First-time Migrates:                    0
  Requeued Migrates:                      0
  Queued Directory Promotes:              0
```

```
    Last Scan Statistics:
      Scan Started:                               Wed Apr  4 03:41:53 2007
      Scan Completed:                             Wed Apr  4 03:55:20 2007
      Elapsed Time:                               00:13:27
      Scan Report:                                docsPlc_20070404034142.rpt
      Number of Files Scanned:                    1043
      Number of Directories Scanned:              305
      Number of Files in Fileset:                 73
      Number of Files Migrated:                   68
      Size of Files Migrated:                     58M (58M on source)
      Number of Directories Promoted:             0
      Number of Failed Migrations:                0
      Number of Failed Directory Promotes:        0


Namespace:        medarcv
Volume:           /rcrds

  Share Farm:       medFm

                 Placement Frequency   Freespace Status
     Share Name   Count/Total           Free   Size   Pct Free
     -----------  --------------------  ----------------------
     charts       1   / 49       2  %   15G     16G     93 %
     rx           48  / 49       97 %   74G     74G     99 %


  New File Placement Rule:  medFm

    Configuration:
      Constrain Files:                            No
      Constrain Directories:                      No
      Balance Mode:                               Latency
      Maintain Freespace:                         1G
      Auto Migrate:                                100M

      State:                                       Enabled

    Status:
      Volume Scan Status:                          Complete
      File Migration Status:                       Complete
      New File Placement Status:                   Enabled

    Cumulative Statistics:
      Total Files Migrated:                        0
      Total Directories Promoted:                  0
      Total Failed Migrations:                     0
      Total Failed Directory Promotes:             0
      Total Retried Migrations:                    0
      Total Canceled Migrations:                   0
      Total Hard Links Skipped:                    0
      Total Files Placed Inline:                   0
      Total File Renames Processed Inline:         0
      Total Directories Placed Inline:             0
      Total Directory Renames Processed Inline:    0
      Number of Scans Performed:                   0

    Queue Statistics:
      First-time Migrates:                         0
      Requeued Migrates:                           0
      Queued Directory Promotes:                   0



  Place Rule:       dailyArchive

    Configuration:
      From fileset:                                dayOld (files only)
      Target share:                                bulk
      Schedule:                                  hourly
      Migrate limit:                             0
      Volume Scan:                               Enabled
      Inline Notifications:                      Disabled
      Promote Directories:                       Disabled

      State:                                     Enabled

    Status:
      Volume Scan Status:                        Complete
      File Migration Status:                     Complete
      New File Placement Status:                 Enabled

    Cumulative Statistics:
      Total Files Migrated:                      0
      Total Directories Promoted:                0
      Total Failed Migrations:                   0
      Total Failed Directory Promotes:           0
      Total Retried Migrations:                  0
      Total Canceled Migrations:                 0
      Total Hard Links Skipped:                  0
      Total Files Placed Inline:                 0
      Total File Renames Processed Inline:       0
      Total Directories Placed Inline:           0
      Total Directory Renames Processed Inline:  0
      Number of Scans Performed:                 1

    Queue Statistics:
      First-time Migrates:                       0
      Requeued Migrates:                         0
      Queued Directory Promotes:                 0

    Last Scan Statistics:
      Scan Started:                              Wed Apr  4 03:44:51 2007
      Scan Completed:                            Wed Apr  4 03:44:51 2007
```

```
      Elapsed Time:                           00:00:00
      Scan Report:                            None
      Number of Files Scanned:                93
      Number of Directories Scanned:          25
      Number of Files in Fileset:             0
      Number of Files Migrated:               0
      Size of Files Migrated:                 0 (0 on source)
      Number of Directories Promoted:         0
      Number of Failed Migrations:            0
      Number of Failed Directory Promotes:    0


Volume:             /lab_equipment
Namespace:          insur
Volume:             /claims

  Filename Fileset:  images

    Configuration:
      Name Is:
      Path Is:                                /images/
      Recurse:                                Yes

bstnA6k# ...
```

Focusing on One Namespace

You can add a namespace name to focus on a particular namespace:

show policy *namespace*

where ***namespace*** (optional, 1-30 characters) is the name of a configured namespace (see "Listing All Namespaces" on page 7-3).

This lists all rules in the chosen namespace, and shows the rule order. Rule order is important for any rules that may conflict: if rule 1 contradicts rule 4 for a given file, rule 1 is enforced for that file.

For example, the following command lists the rule and share farm for the “wwmed” namespace:

```
bstnA6k# show policy wwmed

Namespace:        wwmed

  Rule                                                                          Status
Priority   Volume                   Rule                        Vol. Scan          Migration
---------  -------------------------  -------------------------  -------------------------
    1      /acct                      docs2das8                   Volume Paused     Volume
Paused
    2      /acct                     fm1                         Complete          Complete
bstnA6k# ...
```

Showing Details for the Namespace

Add the details keyword for details about the namespace:

show policy *namespace* details

This lists all the volumes in the namespace, with details about the rules and share farms under each volume. Volumes without any policy settings are listed without any details under them.

For example, this command shows details about the “wwmed” namespace:

```
bstnA6k# show policy wwmed details

Namespace:         wwmed
Volume:            /acct

  Share Farm:        fm1

                   Placement Frequency   Freespace Status
     Share Name    Count/Total           Free    Size    Pct Free
     -----------   --------------------  ----------------------
     bills         179 / 767      23 %   19G      34G     55 %
     bills2        41  / 767      5  %   4.4G     4.9G    91 %
     budget        547 / 767      71 %   59G      62G     94 %
  New File Placement Rule:  fm1

    Configuration:
      Constrain Files:                           No
      Constrain Directories:                     No
      Balance Mode:                              Capacity
...

bstnA6k# ...
```

Focusing on One Volume

Add the volume name after the namespace name to focus on the volume:

show policy *namespace volume*

where

namespace (optional, 1-30 characters) is the namespace, and

volume (optional, 1-1024 characters) identifies the volume.

For example, the following command shows the “/acct” volume in the “wwmed” namespace:

```
bstnA6k(gbl)# show policy wwmed /acct

Namespace:         wwmed
Volume:            /acct

  Rule                                       Status
Priority   Rule                    Vol. Scan          Migration
---------  --------------------  ----------------------------------
    1      docs2das8               Volume Paused          Volume Paused
    2      fm1                     Complete               Complete
bstnA6k# ...
```

Showing Details for the Volume

As with namespaces, you can add the details keyword for details about the volume:

show policy *namespace volume* details

This lists details about all the rules and share farms in the volume. For example, this command shows details about the “wwmed~/acct” volume:

```
bstnA6k# show policy wwmed /acct details

Namespace:          wwmed
Volume:             /acct

  Share Farm:         fm1

                    Placement Frequency    Freespace Status
     Share Name     Count/Total            Free    Size    Pct Free
     -----------    --------------------   ----------------------
     bills          179 / 767      23 %    19G     34G     55 %
     bills2         41  / 767      5  %    4.4G    4.9G    91 %
     budget         547 / 767      71 %    59G     62G     94 %

...
```

Focusing on One Share Farm or Rule

After the optional namespace and volume names, use the name of one of the rules or share farms to narrow the focus to one object:

show policy *namespace volume rule-or-share-farm-name*

where

namespace (optional, 1-30 characters) is the namespace,

volume (optional, 1-1024 characters) identifies the volume, and

rule-or-share-farm-name (optional, 1-1024 characters) identifies the rule.

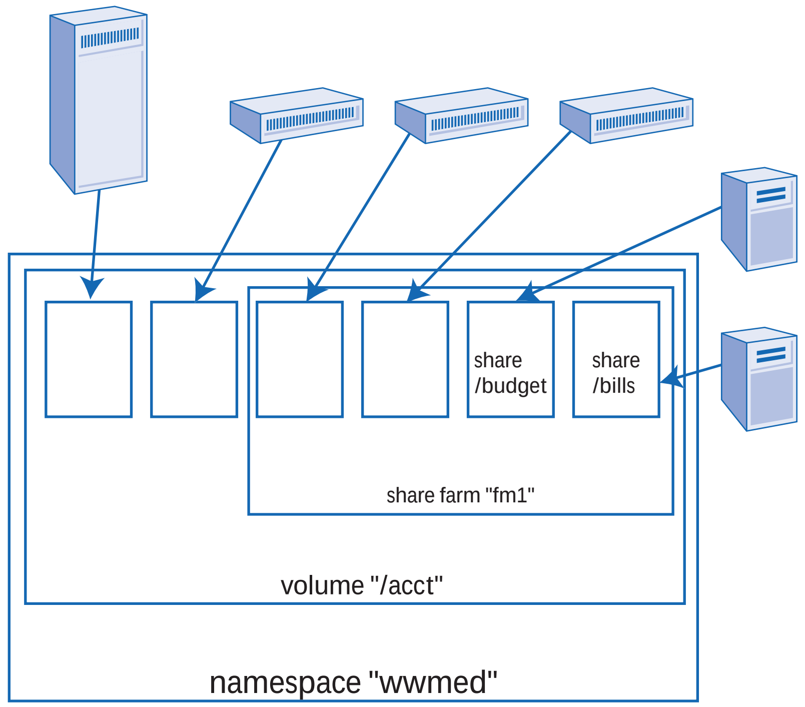
This expands the output to show the full details for the share farm or rule. These details include configuration parameters and usage statistics.

For example, the following command shows the "docs2das8" rule in the "wwmed~/acct" volume:

```
bstnA6k# show policy wwmed /acct docs2das8

Namespace:        wwmed
Volume:           /acct

  Place Rule:       docs2das8

    Configuration:
      From fileset:                             bulky (files only)
      Target share:                             bills
      Report:                                   docsPlc, Verbose
      Migrate limit:                            50G
      Volume Scan:                              Enabled
      Inline Notifications:                     Enabled
      Promote Directories:                      Disabled

      State:                                    Enabled

    Status:
      Volume Scan Status:                       Complete
      File Migration Status:                    Complete
      New File Placement Status:                Enabled

    Cumulative Statistics:
      Total Files Migrated:                     68
      Total Directories Promoted:               0
      Total Failed Migrations:                  0
      Total Failed Directory Promotes:          0
      Total Retried Migrations:                 240
      Total Canceled Migrations:                0
      Total Hard Links Skipped:                 0
      Total Files Placed Inline:                 39
      Total File Renames Processed Inline:       0
      Total Directories Placed Inline:           0
      Total Directory Renames Processed Inline:  0
      Number of Scans Performed:                 1

    Queue Statistics:
      First-time Migrates:                       0
      Requeued Migrates:                         0
      Queued Directory Promotes:                 0

    Last Scan Statistics:
      Scan Started:                              Wed Apr  4 03:41:53 2007
      Scan Completed:                            Wed Apr  4 03:55:20 2007
      Elapsed Time:                              00:13:27
      Scan Report:                               docsPlc_20070404034142.rpt
      Number of Files Scanned:                   1043
      Number of Directories Scanned:             305
      Number of Files in Fileset:                73
      Number of Files Migrated:                  68
      Size of Files Migrated:                    58M (58M on source)
      Number of Directories Promoted:            0
      Number of Failed Migrations:               0
      Number of Failed Directory Promotes:       0


bstnA6k# ...
```

Adding a Share Farm

You configure your usage-balancing policies in a *share farm*. A share farm is a group of shares in a volume. You can apply file-distribution rules to a share farm, with the aim of balancing the usage of its back-end shares. Instructions for setting these rules appear later in this chapter.



A volume can contain one or more share farms. From gbl-ns-vol mode, use the share-farm command to add a share farm to the volume:

share-farm *name*

where ***name*** (1-64 characters) is the name you choose for the share farm.

This puts you into gbl-ns-vol-sfarm mode, where you can identify the shares in the farm.

For example, the following command sequence creates an empty share farm, “fm1:”

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share-farm fm1
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~fm1])# ...
```

Adding a Share to the Farm

The next step in creating a share farm is to add a share to the farm. The share farm can hold multiple shares. Use the share command to add one:

share *name*

where ***name*** (1-64 characters) identifies the share.

For example, the following command sequence adds two shares, “budget” and “bills,” to the share farm named “fm1:”

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share-farm fm1
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~fm1])# share bills
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~fm1])# share budget
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~fm1])# ...
```

Setting a Placement Weight for a Share

Share-farm balancing policies can use different “weights” for each share. A share’s weight determines how many newly-created files it can handle. If you configure round-robin balancing in a share farm (described below), the ARX uses the weights to determine the proportional number of files to assign to each share.

A share weight can be any number between 0 and 100. The weight of each share is compared to the weights of the other shares to determine the number of new files that each should get. If one share has a weight of 50 and two others have weights of 25 each, the first share is tasked with twice as many new files as the other two.

The default weight is 1. From gbl-ns-vol-sfarm mode, use the weight clause with the share command to set the share's weight:

share *name* weight *weight*

where

name (1-64 characters) identifies the share, and

weight (0-100) is the weight of the share, relative to the weights you set for other shares in the same farm. A 0 (zero) makes the share ineligible for new files.

For example, the following command sequence sets the weight for two shares, "budget" and "bills," in the share farm named "fm1:"

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share-farm fm1
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~fm1])# share bills weight 2
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~fm1])# share budget weight 6
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~fm1])# ...
```

Removing a Share from a Share Farm

Use the no form of the share command to remove a share from the current share farm:

no share *name*

where ***name*** (1-64 characters) identifies the share to be removed.

For example, the following command sequence removes the "accounts-payable" share from the share farm named "fm1:"

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share-farm fm1
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~fm1])# no share accounts-payable
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~fm1])# ...
```

Auto Migrating Existing Files

You can configure an *auto-migrate* policy to migrate files off of a share that is low on free space. The files migrate to shares that are *not* low on free space, if there are any such shares in the same share farm. From gbl-ns-vol-sfarm, use the auto-migrate command to trigger an auto-migrate policy:

auto-migrate *freespace*{k|M|G|T}

where:

freespace (1-18,446,744,073,709,551,615) is the amount of free space to maintain on each share, and

k|M|G|T chooses the unit of measure: **k**ilobytes, **M**egabytes, **G**igabytes, or **T**erabytes. All values are base-2; e.g., a kilobyte is 1,024 bytes and a megabyte is 1,048,576 bytes.

For example, the auto-migrate command in the following command sequence ensures that if any share the ‘fm1’ share farm drops below two gigabytes of free space, the share farm migrates some of its files to emptier shares:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share-farm fm1
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~fm1])# auto-migrate 2G
```

Disabling Auto Migration

The auto-migrate policy is disabled by default. Use the no auto-migrate command to return to this default:

no auto-migrate

For example, the no auto-migrate command in the following command sequence disables automatic file migration in the ‘medFm’ share farm.

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# share-farm medFm
bstnA6k(gbl-ns-vol-sfarm[medarcv~/rcrds~medFm])# no auto-migrate
bstnA6k(gbl-ns-vol-sfarm[medarcv~/rcrds~medFm])# ...
```

Balancing New Files Based on Free Space

New files, created by the volume's clients, are distributed round-robin amongst the shares in the share farm. For example, consider a share farm with shares s1 and s2: the first new file goes to s1, the second goes to s2, the third goes to s1, and so on. This is done without regard to free space on each share.

You can configure the share farm to assign new files based on the current free space at each share. (The ARX gets its free-space numbers from each share every 60 seconds.) This algorithm assigns new files based on the relative free space at each share: for example, if s1 has two gigabytes of free space and s2 has one gigabyte of free space, s1 is assigned twice as many new files as s2.

Note

This balancing of new files cannot be intelligent about the size of the files because they are distributed at creation time; a file's size is always zero at the moment it is created. After the file is written and its size is established, the auto-migrate rule can migrate the file to an emptier share as needed.

To balance new-file distribution based on free space, use the balance capacity command from gbl-ns-vol-sfarm mode:

balance capacity

For example, the following command sequence distributes new files in the 'fm1' share farm based on relative free space (capacity) at each share:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share-farm fm1
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~fm1])# balance capacity
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~fm1])# ...
```

Based on Latency (Bandwidth)

The NSM continuously updates its measure of the average *latency* (round-trip packet time) between its ports and each share. A low latency for a share indicates high currently-available bandwidth at the share. You can use the balance command to distribute new files based on latency measures instead of free-space measures. For example, consider a share farm where one share, s1, has an average latency that is three times faster than the latency to s2: this algorithm would send three times as many files to s1 as s2. The share with the lowest latency gets the most new files.

To distribute more new files to shares with lower latency (and therefore more bandwidth), use the balance latency command:

balance latency

For example, the following command sequence configures the 'medFm' share farm to distribute its new files based on the current latency at each share:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# share-farm medFm
bstnA6k(gbl-ns-vol-sfarm[medarcv~/rcrds~medFm])# balance latency
bstnA6k(gbl-ns-vol-sfarm[medarcv~/rcrds~medFm])# ...
```

Based on Administrative Weights

You can also choose to use the weighted-round-robin method of distributing new files, using the weights that you set when you add each share to the share farm (as shown in "Setting a Placement Weight for a Share" on page 12-16). This is the default new-file-placement policy for a new share farm. The number of new files assigned to each share is based on the weight of the share, relative to all the other share weights: for example, if s1 has a weight of 15 and s2 has a weight of 60, s2 gets four times as many new files as s1.

To place the new files on the shares based on their relative weights, use the balance round-robin command:

balance round-robin

For example, the following command sequence causes the share farm to assign twice as many files to the 'back1' share as the 'back2' share:

```
prtlndA1k(gbl)# namespace nemed
```

```
prtlndA1k(gbl-ns[nemed])# volume /acctShdw
prtlndA1k(gbl-ns-vol[nemed~/acctShdw])# share-farm farm1
prtlndA1k(gbl-ns-vol-sfarm[nemed~/acctShdw~farm1])# share back1 weight 20
prtlndA1k(gbl-ns-vol-sfarm[nemed~/acctShdw~farm1])# share back2 weight 10
prtlndA1k(gbl-ns-vol-sfarm[nemed~/acctShdw~farm1])# balance round-robin
prtlndA1k(gbl-ns-vol-sfarm[nemed~/acctShdw~farm1])# ...
```

Maintaining Minimum Free Space

By default, the share farm stops distributing new files to any share whose free space falls below one gigabyte. From gbl-ns-vol-sfarm mode, use the maintain-free-space command to change this free-space threshold:

maintain-free-space *free-space*{k|M|G|T}

where:

free-space (1-18,446,744,073,709,551,615) is the free space that is required on every share.

k|M|G|T chooses the unit of measure for the free space: **k**ilobytes, **M**egabytes, **G**igabytes, or **T**erabytes. All values are base-2; e.g., a kilobyte is 1,024 bytes and a megabyte is 1,048,576 bytes. No space is allowed between the number and this unit of measure: **20G** is correct, **20 G** is not.

Caution

Set this free space at least as high as the free-space number for auto-migrate. If this number is lower, the share may continuously *thrash*, placing new files onto the share and auto-migrating them off again.
For example, consider a share farm that places new files onto its shares until they are down to 1G of free space, but auto migrates until its shares have at least 2G of free space. New files push the free space down toward 1G, but then the auto-migrate rule moves files away until the free space rises back toward 2G.

For example, the following command sequence preserves at least 2 Gigabytes of free space for shares in the ‘fm1’ share farm:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share-farm fm1
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~fm1])# maintain-free-space 2G
```

```
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~fm1])# ...
```

New-File Placement When All Shares Reach the Free Space Threshold

If all shares fill up to their maintain-free-space measures, the share farm distributes each new file to the same share as its parent directory.

Disabling the Free-Space Threshold

You can allow the balance rule to continue placing new files on shares that are close to filling up. Use the no form of the maintain-free-space command to disable the maintaining free space feature.

no maintain-free-space

For example:

```
prtlndA1k(gbl)# namespace nemed
prtlndA1k(gbl-ns[nemed])# volume /acctShdw
prtlndA1k(gbl-ns-vol[nemed~/acctShdw])# share-farm farm1
prtlndA1k(gbl-ns-vol-sfarm[nemed~/acctShdw~farm1])# no maintain-free-space
prtlndA1k(gbl-ns-vol-sfarm[nemed~/acctShdw~farm1])# ...
```

Constraining New Files

Some installations may prefer to retain files on the same back-end shares as their parent directories, at least until it is determined that the files must be auto-migrated to free up space on the share. From gbl-ns-vol-sfarm mode, you can use the constrain-files command to keep any new file in the same share as its parent directory:

constrain-files

This is the opposite of the new-file balance policy described above; if you constrain new files, you cancel the effects of any balance command that is in force.

For example:

```
bstnA6k(gbl)# namespace ns2
bstnA6k(gbl-ns[ns2])# volume /usr
bstnA6k(gbl-ns-vol[ns2~/usr])# share-farm fm4
```

```
bstnA6k(gbl-ns-vol-sfarm[ns2~/usr~fm4])# constrain-files
bstnA6k(gbl-ns-vol-sfarm[ns2~/usr~fm4])# ...
```

Distributing New Files

Use the no form of the constrain-files command to balance new files in the current share farm. With this (default) setting, the ARX uses the new-file balancing algorithm set with one of the balance commands:

no constrain-files

For example:

```
bstnA6k(gbl)# namespace ns2
bstnA6k(gbl-ns[ns2])# volume /usr
bstnA6k(gbl-ns-vol[ns2~/usr])# share-farm fm4
bstnA6k(gbl-ns-vol-sfarm[ns2~/usr~fm4])# no constrain-files
bstnA6k(gbl-ns-vol-sfarm[ns2~/usr~fm4])# ...
```

Constraining New Directories

By default, an enabled share farm distributes all new directories by following the balance rule. You can instead constrain any new directory to the same share as its parent. The constrain-directories command imposes this constraint.

This only makes sense in a share farm where you constrain new files, too. If new-file balancing is enabled and a client creates several new files in the same directory, that directory must be copied onto each share that receives any of the new files. This is called *striping* the directory. The constrain-directories command cannot prevent this; in this case, it only determines which share has the *master* copy of the directory.

The auto-migrate rule generally causes less striping than the balance rule, so it can co-exist with constrained directories.

From gbl-ns-vol-sfarm mode, use the constrain-directories command to constrain new directories to the same share as their parent directories:

constrain-directories

For example, the following command sequence causes all new directories in the ns2~/usr~fm4 share farm to remain in the same shares as their parent directories:

```
bstnA6k(gbl)# namespace ns2
bstnA6k(gbl-ns[ns2])# volume /usr
bstnA6k(gbl-ns-vol[ns2~/usr])# share-farm fm4
bstnA6k(gbl-ns-vol-sfarm[ns2~/usr~fm4])# constrain-directories
bstnA6k(gbl-ns-vol-sfarm[ns2~/usr~fm4])# ...
```

Constraining Directories Below a Certain Depth

You can apply the directory constraint to any level in the volume's directory tree. For example, consider a volume called /var that gets three new child directories, /usr, /log, and /bin: if you constrain all directories below the first level of this tree, the share farm can distribute those directories to any share with available space (as guided by the balance rule). The constraint applies to directories below that first level: any directory created inside /usr, /log, or /var is constrained to the same share as its parent directory.

To constrain directories below a certain depth, add the optional below-depth argument to the constrain-directories command:

constrain-directories below *depth*

where ***depth*** (0-100) is the highest level in the volume's directory tree where balance is still enforced. The directory trees below that level are constrained to the same share as their parent directories. This defaults to 0, which constrains all directories in the volume.

For example, the following command sequence causes all new directories in the ns2~/usr~fm4 share farm to be constrained to their parent directories when the new directory is below the first subdirectory level.

```
bstnA6k(gbl)# namespace ns2
bstnA6k(gbl-ns[ns2])# volume /usr
bstnA6k(gbl-ns-vol[ns2~/usr])# share-farm fm4
bstnA6k(gbl-ns-vol-sfarm[ns2~/usr~fm4])# constrain-directories below 1
bstnA6k(gbl-ns-vol-sfarm[ns2~/usr~fm4])# ...
```

Given a depth of 'below 1', if the ns2~/usr~fm4 share farm has shares s1, s2, and s3 with the same free space and capacity, and you have created root directories /a, /b, and /c, these directories are spread evenly across shares s1, s2, and s3 one share at a time. Any directory created in /a, /b, or /c is placed on the same share as its parent directory.

Not Constraining Directories

Use no constrain-directories to remove directory placement restrictions and have new directories distributed as directed by one of the balance commands.

no constrain-directories

For example:

```
bstnA6k(gbl)# namespace ns2
bstnA6k(gbl-ns[ns2])# volume /var
bstnA6k(gbl-ns-vol[ns2~/var])# share-farm fm2
bstnA6k(gbl-ns-vol-sfarm[ns2~/var~fm2])# no constrain-directories
bstnA6k(gbl-ns-vol-sfarm[ns2~/var~fm2])# ...
```

Distributing When a Share is Unavailable

When new-file or new-directory distribution is constrained, a user may get an error if one share in the farm is unavailable (offline or otherwise unusable). The share farm is constrained to creating the new file or directory on the same share as its parent, so the user gets a create error if that share goes offline. The unavailable share is removed from the new-file-balancing rotation automatically, unless a constrain command is used.

Enabling All Share-Farm Rules

The final step in configuring a share farm is to enable its rules. By default, the share-farm and its rules are disabled and ignored by policy software. From gbl-ns-vol-sfarm mode, use the enable command to enable the share farm:

enable

For example, the following command sequence enables the ‘fm1’ share farm:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share-farm fm1
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~fm1])# enable
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~fm1])# ...
```

Stopping All Share-Farm Rules

You can stop all auto migrations and/or new-file balancing on a share farm by disabling it. This reverts all shares to standard behavior; no auto migrations as free space gets low on a share, and any new file or directory is created on the same share as its parent. To do this, use the no enable command from gbl-ns-vol-sfarm mode.

no enable

For example, the following command sequence stops all rules in the 'fm2' share farm:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share-farm tst
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~tst])# no enable
bstnA6k(gbl-ns-vol-sfarm[wwmed~/acct~tst])# ...
```

Removing a Share Farm

You cannot remove a share farm if any rules use it as a source or target. (Later sections and chapters explain how to configure rules that use share farms in this way.) Remove all such rules before removing the share farm; use show policy to find them, as instructed in "Showing All Policy Rules" on page 12-2.

From gbl-ns-vol mode, use the no share-farm command to remove a share farm:

no share-farm *name*

where ***name*** (1-64 characters) identifies the share farm to be removed.

This does not affect client access to the farm's shares.

For example, the following command sequence removes the "san14luns4-8" share farm from the "/acct" volume:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# no share-farm san14luns4-8
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Creating a Schedule

Several policy rules use a *schedule*, which determines when (and how frequently) a rule runs. Each rule can have a unique schedule. Conversely, several rules can share the same schedule. From gbl mode, use the schedule command to create a policy schedule:

schedule *name*

where ***name*** (1-64 characters) is the name that you assign to the schedule,

For example, the following command creates a schedule named “hourly:”

```
bstnA6k(gbl)# schedule hourly
bstnA6k(gbl-schedule[hourly])# ...
```

Setting the Interval

The next step in creating a schedule is to set the schedule’s interval (such as “every 5 minutes,” “every 3 hours,” or “every Monday”). From gbl-schedule mode, use the every command to set the schedule’s interval:

every *number* { minutes | hours | days | weeks | months | quarters | years }

or

every *day-of-week*+

where you can choose between two formats:

- **every *number unit***, where the *number* is an integer and *unit* is minutes, hours, days, weeks, months, or years. For example, **every 15 minutes**, **every 3 weeks**, **every 4 days**, or **every 1 years**.
- **every *day-of-week***, where the *day-of-week* is the full name of the day in lower case. For example, **every monday**, **every wednesday**, or **every sunday**. You can enter more than one to run the schedule more than once per week; for example, **every monday wednesday friday** runs three times per week.

The following sample-command sequence configures the “hourly” schedule to fire once per hour:

```
bstnA6k(gbl)# schedule hourly
bstnA6k(gbl-schedule[hourly])# every 1 hours
bstnA6k(gbl-schedule[hourly])# ...
```

Setting the Duration (optional)

The next step in creating a schedule is to set a duration. The duration is the amount of time that a rule can run. The duration is applied every time the schedule fires: if you set a 5-minute duration for the schedule, each rule that uses the schedule has 5 minutes to run every time it runs.

At the end of the duration, a rule with this schedule stops any of its volume scans and/or migrations. If a client changes a file so that it should migrate according to the rule, the policy engine caches the migration request. All migrations resume the next time the rule runs. On the other hand, the rule continues to direct *new* files and directories to their configured target shares; no migrations are necessary, so this is not controlled by the schedule or its duration.

From gbl-schedule mode, use the duration command to set the duration:

```
duration time-interval
```

where ***time-interval*** determines the duration of each scheduled event. Express the *time-interval* in *hh*:*mm*:*ss* format; for example, **duration 00:15:00** for 15 minutes, or **duration 06:00:00** for six hours.

For example, the following command sequence configures the “hourly” schedule to run for 5 minutes each time it fires:

```
bstnA6k(gbl)# schedule hourly
bstnA6k(gbl-schedule[hourly])# duration 00:05:00
bstnA6k(gbl-schedule[hourly])# ...
```

Removing the Duration

The default is no limit on the duration of a rule execution. To remove the duration and return to this default, use no duration:

```
no duration
```

For example, the following command sequence removes the duration from the “hourly” schedule:

```
bstnA6k(gbl)# schedule hourly
bstnA6k(gbl-schedule[hourly])# no duration
bstnA6k(gbl-schedule[hourly])# ...
```

Setting the Start Time (optional)

A schedule’s *start time* determines the start of each interval: if a daily schedule has a start time of 2:42 PM, the schedule will fire at 2:42 PM every day. By default, a schedule’s start time is the time that it is configured. You can optionally use the start command to set a different start time and date:

start *date*:*time*

where the colon (**:**) separates the date from the time.

date defines the start date for the schedule. Use the format ***mm*/*dd*/*yyyy*** (for example, **04/01/2003** for April 1, 2003, or **11/20/2005** for November 20, 2005).

time defines the start time. Use the format ***HH*:*MM*:*SS*** (for example, **12:00:00** for noon, or **17:00:00** for 5 PM).

For example, the following command sequence sets the hourly schedule to start at 2 AM on October 24, 2004:

```
bstnA6k(gbl)# schedule hourly
bstnA6k(gbl-schedule[hourly])# start 10/24/2004:02:00:00
bstnA6k(gbl-schedule[hourly])# ...
```

Starting Now

Use the no start command to erase the previously-configured start time and start the schedule now:

no start

For example:

```
bstnA6k(gbl)# schedule daily
```

```
bstnA6k(gbl-schedule[daily])# no start
bstnA6k(gbl-schedule[daily])# ...
```

Showing All Schedules

To list all schedules on the switch, use the show policy schedule command:

show policy schedule

This shows each schedule’s configuration parameters as well as the time of the next scheduled run.

For example:

```
bstnA6k(gbl)# show policy schedule

  Schedule:          hourly
  Start Time:        Sun Oct 24 01:00:00 2004
  Previous Run:      Wed Apr  4 05:00:00 2007
  Runs Next:         Wed Apr  4 06:00:00 2007
  Interval:          1 hours

  Schedule:          daily4am
  Start Time:        Sun Sep  4 03:00:00 2005
  Previous Run:      Wed Apr  4 03:00:00 2007
  Runs Next:         Thu Apr  5 03:00:00 2007
  Interval:          1 days
  Duration:          02:00:00
  End Time:          Thu Apr  5 05:00:00 2007

  Schedule:          backupWindow
  Start Time:        Sun Nov 12 13:00:00 2006
  Previous Run:      Tue Apr  3 13:00:00 2007
  Runs Next:         Wed Apr  4 13:00:00 2007
  Interval:          1 days
  Duration:          04:00:00
  End Time:          Wed Apr  4 17:00:00 2007
bstnA6k(gbl)# ...
```

Showing One Schedule

To focus on a single schedule, add the desired schedule name to the command:

show policy schedule *name*

where ***name*** (1-64 characters) identifies the schedule to show.

For example, this shows the “hourly” schedule:

```
bstnA6k(gbl)# show policy schedule hourly

  Schedule:           hourly
  Start Time:         Sun Oct 24 01:00:00 2004
  Previous Run:       Wed Apr  4 05:00:00 2007
  Runs Next:          Wed Apr  4 06:00:00 2007
  Interval:           1 hours

bstnA6k(gbl)# ...
```

Removing a Schedule

Use the no form of the schedule command to remove a schedule from the switch configuration:

no schedule *name*

where ***name*** (1-64 characters) identifies the schedule to be removed.

For example, the following command sequence shows all schedules and deletes the schedule named “daily4am:”

```
bstnA6k(gbl)# show policy schedule

  Schedule:           hourly
  Start Time:         Sun Oct 24 01:00:00 2004
  Previous Run:       Wed Apr  4 05:00:00 2007
  Runs Next:          Wed Apr  4 06:00:00 2007
  Interval:           1 hours

  Schedule:           daily4am
  Start Time:        Sun Sep  4 03:00:00 2005
  Previous Run:      Wed Apr  4 03:00:00 2007
  Runs Next:         Thu Apr  5 03:00:00 2007
  Interval:          1 days
  Duration:          02:00:00
  End Time:          Thu Apr  5 05:00:00 2007

  Schedule:          backupWindow
  Start Time:        Sun Nov 12 13:00:00 2006
  Previous Run:      Tue Apr  3 13:00:00 2007
  Runs Next:         Wed Apr  4 13:00:00 2007
  Interval:          1 days
  Duration:          04:00:00
  End Time:          Wed Apr  4 17:00:00 2007

bstnA6k(gbl)# no schedule daily4am
bstnA6k(gbl)# ...
```

Pausing All Rules in a Volume

There are occasions when it is helpful to stop all file migrations, such as times scheduled for backups (see "Backing Up a Volume's Files" on page 2-1 of the *CLI Maintenance Guide*). You can use the policy pause command from priv-exec mode to immediately suspend all policy rules in a given volume:

policy pause *namespace vol-path*

where

namespace (1-30 characters) is the name of a namespace.

vol-path (1-1024 characters) identifies the volume.

This pauses all of the volume's rules, so that they stop all volume scans and migrations. Clients may change files or directories so that they match a rule and therefore should be migrated; these migrations are queued until policy processing is resumed later. All file-placement rules continue to direct *new* files and directories to their configured storage. (New objects are created at the correct share, so no migrations are necessary.) Pausing has the same effect as an expired duration on a rule's schedule; recall "Setting the Duration (optional)" on page 12-28.

For example, the following command pauses all rules in the "wwmed~/acct" volume:

```
bstnA6k(gbl)# end
bstnA6k# policy pause wwmed /acct
bstnA6k# ...
```

Resuming all Policies in a Volume

You can use the priv-exec form of no policy pause to immediately resume all rule processing in a volume. This restarts rules that were paused from the priv-exec command. This has no effect on scheduled policy pauses, described below.

no policy pause *namespace vol-path*

For example, the following command sequence un-pauses policy in the "wwmed~/acct" volume:

```
bstnA6k(gbl)# end
bstnA6k# no policy pause wwmed /acct
bstnA6k# ...
```

Pausing on a Schedule

Some installations want to schedule “off hours” for file migrations; for example, you may want to pause all migrations during regularly scheduled backup windows. You can create a schedule (as described earlier) to define the off hours, then pause a volume according to that schedule. This reduces resource contention between clients and the policy engine. From gbl-ns-vol mode, use the policy pause command to pause the current volume’s rules on a schedule:

policy pause *schedule*

where ***schedule*** (1-64 characters) is the name of a schedule. Use show policy schedule for a list of configured schedules (see “Showing All Schedules” on page 12-30).

Each time the schedule fires, the volume stops all volume scans and migrations as described above. The schedule must have a limited duration, or policy never resumes; for instructions on setting a duration on a schedule, see “Setting the Duration (optional)” on page 12-28.

For example, the following command sequence assigns the “backupWindow” schedule to the “wwmed~/acct” volume:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# policy pause backupWindow
bstnA6k(gbl-ns-vol[wwmed~/acct])# ...
```

Removing the Policy-Pause Schedule

From gbl-ns-vol mode, you can use the no policy pause command to stop pausing policy on a schedule:

no policy pause

This has no effect on a policy pause command issued from priv-exec mode.

For example, the following command sequence stops all scheduled-policy pauses in the “medarcv~/rcrds” volume:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
```

```
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# no policy pause
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

Draining One or More Shares

You can move all files from one share (or share farm) to one or more other shares in the same volume. A placement rule accomplishes this, and prevents any new files from being created on the source share(s). This is a method to prepare one or more shares for removal without affecting any clients.

From gbl-ns-vol mode, use place-rule to create a new placement rule:

place-rule *name*

where ***name*** (1-64 characters) is a name you choose for the rule.

This puts you into gbl-ns-vol-plc mode, where you must choose a source share or share farm, set the storage target for the files, and enable the rule. There are also some optional commands you can invoke from this mode.

For example, the following command sequence creates an empty placement rule, “emptyRH,” for the namespace, “wwmed:”

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule emptyRH
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# ...
```

Identifying the Source Share(s)

The next step in configuring a placement rule is to identify the source share or shares. The placement rule places all files from the source share(s) onto the target storage. From gbl-ns-vol-plc mode, use one of two source commands to specify the source share(s):

source share *share-name*

where **share *share-name*** (1-64 characters) identifies a single source share, or

source share-farm *share-farm-name*

where ***share-farm-name*** (1-64 characters) is a group of shares in a share farm. This drains all shares in the share farm.

Use the show global-config namespace command to see the shares or share farms in each volume: see "Showing Namespace Configuration" on page 7-25.

For example, the following command set selects the "it5" share:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule emptyRH
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# source share it5
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# ...
```

Choosing the Target Storage

The next step in configuring a placement rule is to choose the target storage for the share's files. You can choose one target: another share or share farm in the current volume. From gbl-ns-vol-plc mode, use one of two target rules to set the share's storage target:

target share *share-name*

where ***share-name*** (1-64 characters) is a share from the current volume.

target share-farm *share-farm-name*

where ***share-farm-name*** (1-64 characters) is a share farm within the current volume.

For example, the following command sequence selects a share farm, "fm1," as the target for the "emptyRH" placement rule:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule emptyRH
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# target share-farm fm1
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# ...
```

As another example, the following command sequence selects a share, "nas3," as the new home for all files in the source share:

```
bstnA6k(gbl)# namespace archives
bstnA6k(gbl-ns[archives])# volume /etc
bstnA6k(gbl-ns-vol[archives~/etc])# place-rule move2nas
bstnA6k(gbl-ns-vol-plc[archives~/etc~move2nas])# target share nas3
bstnA6k(gbl-ns-vol-plc[archives~/etc~move2nas])# ...
```

Balancing Capacity in a Target Share Farm

When a share farm is a file-placement target, the first configured share in the farm is the default share for placed files. Most files are placed on the same share as their parent directory, but a file defaults to the first share if its parent directory is outside the share farm. The first share in the farm can therefore take a heavier file burden over time.

To migrate files off of any share that is running low on free space, you can configure *auto migration* for the share farm. Refer back to "Auto Migrating Existing Files" on page 12-18 to configure auto migration.

Applying a Schedule (optional)

You can optionally run the placement rule at a later start time, set by a schedule. By default, placement rule runs as soon as you enable it, empties the share(s) of all files, and then keeps the share empty. If you limit the amount of data that can migrate in each run (as described later), the schedule can stagger the migration over multiple, smaller runs.

Use the gbl schedule command to create a schedule; refer back to "Creating a Schedule" on page 12-27 for details. To apply a schedule to the placement rule, use the schedule command in gbl-ns-vol-plc mode:

schedule *name*

where ***name*** (1-64 characters) identifies the schedule. Use the show policy command to list all schedules.

For example, the following command sequence applies a schedule to the "move2nas" rule:

```
bstnA6k(gbl)# namespace archives
bstnA6k(gbl-ns[archives])# volume /etc
bstnA6k(gbl-ns-vol[archives~/etc])# place-rule move2nas
bstnA6k(gbl-ns-vol-plc[archives~/etc~move2nas])# schedule fri5pm
bstnA6k(gbl-ns-vol-plc[archives~/etc~move2nas])# ...
```

Removing the Schedule

Use no schedule to make the placement rule run when enabled:

no schedule

For example:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule emptyRH
```

```
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# no schedule
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# ...
```

Limiting Each Migration (optional)

You can use the limit-migrate command to put a ceiling on the amount of data migrated. The policy engine migrates files until it meets this limit; it stops migrating as soon as it discovers that the next file would exceed the limit.

limit-migrate *size*[k|M|G|T]

where

size (1-18,446,744,073,709,551,615) is the size, and

k|M|G|T (optional) is the units; **k**ilobytes (1024 bytes), **M**egabytes (1024*1024 bytes), **G**igabytes (1024*1024*1024), or **T**erabytes (1024*1024*1024*1024). The default is bytes.

This limit applies to every run of the placement rule, so you can use it in conjunction with a schedule to migrate a limited amount of data during off-hours. For example, you can allow a limit of 20 Gigabytes and a daily schedule that runs at midnight. If the source share contains 100 Gigabytes of data, it would migrate over five nights, 20G per night.

For a placement rule without a schedule, this limit applies to the one-and-only run of the rule. If the share exceeds this limit, the left-over files remain on the source share indefinitely. New files are always created at the target share(s), so they are not blocked by this limit.

For example, the following command sequence sets a 10G limit on the “move2nas” rule, which only runs on Friday afternoons:

```
bstnA6k(gbl)# namespace archives
bstnA6k(gbl-ns[archives])# volume /etc
bstnA6k(gbl-ns-vol[archives~/etc])# place-rule move2nas
bstnA6k(gbl-ns-vol-plc[archives~/etc~move2nas])# schedule fri5pm
bstnA6k(gbl-ns-vol-plc[archives~/etc~move2nas])# limit-migrate 10G
bstnA6k(gbl-ns-vol-plc[archives~/etc~move2nas])# ...
```

Removing the Limit

By default, a placement rule migrates until the source share is drained. Use the no limit-migrate command to return to this default:

no limit-migrate

For example, the following command sequence removes any migration limit in the “mvTars” rule:

```
bstnA6k(gbl)# namespace archives
bstnA6k(gbl-ns[archives])# volume /home
bstnA6k(gbl-ns-vol[archives~/home])# place-rule mvTars
bstnA6k(gbl-ns-vol-plc[archives~/home~mvTars])# no limit-migrate
bstnA6k(gbl-ns-vol-plc[archives~/home~mvTars])# ...
```

Configuring a Progress Report (optional)

A progress reports shows all the milestones and results of the file-placement execution. We recommend this to verify that all files were successfully removed from the share. From gbl-ns-vol-plc mode, use the report command to generate this report:

report *prefix*

where ***prefix*** (1-64 characters) is the prefix to be used for the report. The report has a name in the following format: *prefix_YearMonthDayHourMinute*.rpt (for example, clrOldFiler_200403031200.rpt for a report with the “clrOldFiler” prefix).

Use the show reports command for a list of all reports. Use show reports *report-name* to read the report, show reports status *report-name* to get a one-line summary of the report, grep to search through the full report, or tail to tail a report as it is being written.

This only generates a report if at least one existing file is migrated. It does not report on placement of new files, created by clients through the VIP.

For example, the following command sequence enables a report for the “emptyRH” rule:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
```

```
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule emptyRH
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# report emptyRH_
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# ...
```

Generating a Verbose Report

The placement report is terse by default. You should make the file verbose to give the best-possible chance of diagnosing any problems. To accomplish this, use the optional verbose flag at the end of the report command:

report *prefix* verbose

where ***prefix*** is explained above.

For example, the following command resets “emptyRH” to produce a verbose report:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule emptyRH
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# report emptyRH_ verbose
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# ...
```

Deleting Empty Reports

By default, the placement rule creates a report even if no files are migrated. To avoid the empty report file, use the optional delete-empty flag at the end of the report command:

report *prefix* [verbose] delete-empty

where ***prefix*** and **[verbose]** are explained above.

For example, the following command resets “emptyRH” to delete the report if it is empty:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule emptyRH
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# report emptyRH_ verbose delete-empty
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# ...
```

Disabling Reports

From gbl-ns-vol-plc mode, use no report to prevent the rule from generating a report:

no report

This has no effect after the first (and only) run of the placement rule.

For example, the following command sequence disables reporting for the rule, "mvTars:"

```
bstnA6k(gbl)# namespace archives
bstnA6k(gbl-ns[archives])# volume /home
bstnA6k(gbl-ns-vol[archives~/home])# place-rule mvTars
bstnA6k(gbl-ns-vol-plc[archives~/home~mvTars])# no report
bstnA6k(gbl-ns-vol-plc[archives~/home~mvTars])# ...
```

Retaining Copies of Files on the Share (optional)

Before you enable the rule and start migrating files off of the share, you have the option to retain copies of those files. From gbl-ns-vol-shr mode, use the migrate retain-files command to put this safeguard in place:

migrate retain-files

The file copies go to a hidden directory, "~acopia_msnap," at the root of the share. To restore the files later, you can access the back-end share directly (it is not included in the volume).

Caution

Do not use this command for a share in a share farm. If a share-farm-balancing policy tries to move files off of this share, this placement rule makes copies of them, thereby ensuring that the share's free space never actually changes. The balancing policy will therefore move files off of the share indefinitely, unnecessarily occupying space on other shares.

Use this only as a safeguard when draining all files from a share.

For example, the following command sequence retains all migrated files in the it5 share:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share it5
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~it5])# migrate retain-files
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~it5])# ...
```

Disabling File Retention

To disable the feature that retains copies of migrated files, use the no form of the command:

no migrate retain-files

For example, the following command sequence disables the migration safeguard in the it5 share:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# share it5
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~it5])# no migrate retain-files
bstnA6k(gbl-ns-vol-shr[wwmed~/acct~it5])# ...
```

Enabling the Placement Rule

The final step in configuring a placement rule is to enable it. By default, the rule is disabled and ignored by policy software. Use the enable command to enable the rule:

enable

For example, the following command sequence enables the “emptyRH” rule:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule emptyRH
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# enable
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# ...
```

Tentatively Enabling the Rule

A tentatively-enabled rule is configured to appear in the system logs (syslog) as “tentative,” showing the potential effects of the rule if it was enabled. (The log component, POLICY_ACTION, creates the syslog messages; syslog access and log components are described in the *CLI Maintenance Guide*.) If you configured reporting for the rule (as shown above), the report shows which files would be migrated if the rule was fully enabled. Tentative rules do not change policy enforcement, but they do consume processing time in the policy software. From gbl-ns-vol-plc mode, use the enable tentative command to tentatively enable the placement rule:

enable tentative

Use show logs syslog or grep *pattern* logs syslog to view the syslog.

For example, the following command sequence puts the “emptyRH” rule into a tentative state:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule emptyRH
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# enable tentative
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# ...
```

Disabling the Rule

Disabling the rule removes it from consideration. Use no enable from gbl-ns-vol-plc mode to disable a placement rule:

no enable

For example, the following command sequence disables the “emptyRH” rule:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule emptyRH
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# no enable
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# ...
```

Verifying That All Files Are Removed

You can use a metadata-only report to verify that the share is empty. Use the nsck ... metadata-only command to generate a report about the share (as described in "Focusing on One Share" on page 5-10 of the *CLI Maintenance Guide*), and then use show reports to view the report. For example, the following command sequence generates a report on the "it5" share, proving that it has no files:

```
bstnA6k(gbl)# nsck wwmed report metadata-only share it5
Scheduling report: metadata_only.9.rpt on switch bstnA6k
bstnA6k(gbl)# show reports metadata_only.9.rpt
**** Metadata-Only Report: Started at Wed Sep 21 12:10:59 2005 ****
**** Namespace: wwmed
**** Volume: /acct
**** Path: /acct
**** Share: it5

Share                 Physical Filer
--------------------  ----------------------------------------------------------
[it5                ] 192.168.25.24:/lhome/it5

**** Legend:
****   FL = File: The reported entry is a file.
****   DR = Directory: The reported entry is a directory.
****   LN = Link: The reported entry has a link count greater than one.
****   NL = No Lock: Was unable to lock parent directory during report.
****   CC = NFS case-blind name collision.
****   IC = Name contains invalid CIFS characters.
****   FN = Name appears to be filer-generated.
****   NM = Name contains characters that are not mappable to the NFS encoding.
****   SP = A persistent case collision split is registered in the metadata.
****   SN = A persistent split restricts CIFS access, but no collision exists.

Type            Share                  Path
--------------  --------------------   -------------------------------------------
[   DR       ]  [it5                ]  /multiTier
[   DR       ]  [it5                ]  /loadBalance
```

```
**** Total Files:                      0
**** Total Directories:                2
**** Total Links:                      0
**** Total Locking Errors:             0

**** Total items:                    2
**** Elapsed time:            00:00:00
**** Metadata-Only Report: DONE at Wed Sep 21 12:10:59 2005 ****
```

Removing the Placement Rule

You can remove a placement rule to both disable it and delete its configuration. From gbl-ns-vol mode, use the no form of the place-rule command to remove a placement rule:

no place-rule *name*

where ***name*** (1-64 characters) identifies the rule to be removed.

For example, the following command sequence removes the placement rule, “placeOnSAN19,” from the “medarcv~/rcrds” volume:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# no place-rule placeOnSAN19
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

Removing All Policy Objects from a Namespace

You can use a single command to remove all rules, share farms, and other policy objects from a namespace. The remove namespace command has been described in previous chapters for removing a volume ("Removing a Direct Volume" on page 8-37 or "Removing a Managed Volume" on page 9-71) or an entire namespace ("Removing a Namespace" on page 7-27); add the optional policy-only keyword to remove only the policy objects:

remove namespace *name* policy-only [timeout *seconds*] [sync]

where:

name (1-30 characters) is the name of the namespace,

policy-only is the option to remove only the policy objects,

seconds (optional, 300-10,000) sets a time limit on each of the removal's component operations, and

sync (optional) waits for the removal to finish before returning. With this option, the CLI lists the policy objects as it removes them.

The CLI prompts for confirmation before removing the policy objects. Enter **yes** to continue. This operation generates a report named "removeNs_*namespace*_*date*.rpt" if you omit the sync option.

For example, this command sequence exits to priv-exec mode and then synchronously removes all policy objects from the "insur_bkup" namespace:

```
prtlndA1k(gbl)# end
prtlndA1k# remove namespace insur_bkup policy-only timeout 600 sync

Remove components from namespace 'insur_bkup'? [yes/no] yes
% INFO: Removing service configuration for namespace insur_bkup
% INFO: Removing volume policies for namespace insur_bkup
% INFO: destroy policy insur_bkup /insurShdw
prtlndA1k# ...
```

Removing All Policy Objects from a Volume

The optional volume argument focuses the remove namespace command on one volume:

> **remove namespace *name* volume *volume* policy-only [timeout *seconds*] [sync]**

> where:

>> ***name*** (1-30 characters) is the name of the namespace,
>>
>> ***volume*** (1-1024 characters) is the path name of the volume,
>>
>> **policy-only** is the option to remove only the policy objects, and
>>
>> ***seconds*** (optional, 300-10,000) sets a time limit on each of the removal's component operations, and
>>
>> **sync** (optional) waits for the removal to finish before returning, as described above.

For example, this command sequence exits to priv-exec mode and then removes all policy objects from the "insur~/claims" volume:

```
bstnA6k(gbl)# end
bstnA6k# remove namespace insur volume /claims policy-only

Remove policy components from volume '/claims' in namespace 'insur'? [yes/no] yes
...
```

Migrations in a Multi-Protocol Namespace

Migrations in a multi-protocol (NFS and CIFS) volume have subtle side effects. These side effects compensate for limitations in back-end filers; clients are unaffected in most cases, but this section is offered so that you are aware of any potential issues.

Skip to the next section if your namespace is CIFS-only or NFS-only.

File-Attribute Migrations

The policies in this chapter migrate file attributes along with the files themselves. *File attributes* are permission settings, the name or ID of the user who owns the file, the group or groups who have access to the file, last-access times, named streams, and other external data associated with the file. File-attribute migrations require special semantics in a multi-protocol namespace (that is, a namespace whose shares support both NFS and CIFS access). If your policies are set in a namespace that is NFS-only or CIFS-only, you can skip this section.

A multi-protocol namespace can be backed by a heterogeneous mix of multi-protocol filers (MPFs), possibly from multiple vendors. The ARX passes client requests to these filers, and passes filer responses back to the clients. File attributes, such as file ownership and permission settings, are managed by each filer.

NFS and CIFS have fundamentally different file attributes, and each MPF vendor has unique semantics for reconciling the two. An NFS client can change a file's attributes on Filer X, and Filer X uses its semantics to change the CIFS attributes accordingly. This translation is manifestly inexact, so an MPF from another vendor typically gets a slightly different CIFS translation for the same NFS attributes.

If the ARX migrates the filer from Filer X to Filer Y (made by two different vendors), it takes these different semantics into account. This ensures that the NFS and CIFS file attributes are preserved *as much as possible* as they are migrated from one vendor to another. In some cases, file attributes are interpreted differently on the destination filer. The sections below share the details of attribute migration.

Use these tables along with vendor documentation to determine any implications for your clients.

From a NetApp Filer, UNIX Qtree

The following table shows how file attributes are migrated from a NetApp filer with a UNIX-based Qtree:

Vendor for Destination Filer	UNIX		NTFS	
	Permission Bits	**UID, GID, time stamps, ...**	**Security Descriptor (SD)**	**Named Streams**
NetApp, UNIX Qtree	copied	copied	DOS attributes only (as dictated by destination Qtree) Full SD is derived by destination (NetApp) filer	copied
NetApp, NTFS Qtree	Migrates between these two types of Qtrees are not supported.			
SMB	copied	copied	not copied	not supported by SMB/UNIX
SMB with CIFS ACLs	copied	copied	not copied; derived by destination (SMB) filer	not supported by SMB/UNIX
EMC	all copied *after* SD		derived by NetApp, then copied (overriding any volume ACLs)	not supported by EMC

From a NetApp Filer, NTFS Qtree

The following table shows how file attributes are migrated from a NetApp filer with an NTFS-based Qtree:

Vendor for Destination Filer	UNIX		NTFS	
	Permission Bits	**UID, GID, time stamps, ...**	**Security Descriptor (SD)**	**Named Streams**
NetApp, UNIX Qtree	Migrates between these two types of Qtrees are not supported.			

Vendor for Destination Filer	UNIX		NTFS	
	Permission Bits	UID, GID, time stamps, ...	Security Descriptor (SD)	Named Streams
NetApp, NTFS Qtree	only high (sticky) bits copied; remaining attributes derived by destination (NetApp) filer		copied	copied
SMB	copied	copied	not copied	not supported by SMB/UNIX
SMB with CIFS ACLs	copied	copied	not copied; derived by destination (SMB) filer	not supported by SMB/UNIX
EMC	copied	copied	copied *after* UNIX attributes	not supported by EMC

From an EMC Filer

The following table shows how file attributes are migrated from an EMC filer:

Vendor for Destination Filer	UNIX		NTFS	
	Permission Bits	**UID, GID, time stamps, ...**	**Security Descriptor (SD)**	**Named Streams**
NetApp, UNIX Qtree	copied	copied	DOS attributes only (as dictated by destination Qtree) Full SD is derived by destination (NetApp) filer	not supported by EMC
NetApp, NTFS Qtree	only high (sticky) bits copied; remaining attributes derived by destination (NetApp) filer		copied *after* UNIX high bits are set	
SMB	copied	copied	not copied	
SMB with CIFS ACLs	copied	copied	not copied; derived by destination (SMB) filer	
EMC	copied	copied	copied *after* UNIX attributes	

From a Unix Filer with SMB

The following table shows how file attributes are migrated from an SMB filer that does not support CIFS ACLs:

Vendor for Destination Filer	UNIX		NTFS	
	Permission Bits	**UID, GID, time stamps, ...**	**Security Descriptor (SD)**	**Named Streams**
NetApp, UNIX Qtree	copied	copied	DOS attributes only (as dictated by destination Qtree) Full SD is derived by destination (NetApp) filer	not supported by SMB/UNIX

Vendor for Destination Filer	UNIX		NTFS	
	Permission Bits	**UID, GID, time stamps, ...**	**Security Descriptor (SD)**	**Named Streams**
NetApp, NTFS Qtree	Migrates from SMB (no ACLs) to NetApp/NTFS are not supported.			
SMB	copied	copied	not copied	not supported by SMB/UNIX
SMB with CIFS ACLs	copied	copied	not copied; derived by destination (SMB) filer	
EMC	copied, then reset *after* file is transferred through CIFS		copied	

From a UNIX/SMB Filer with ACLs

The following table shows how file attributes are migrated from an SMB filer that supports NTFS ACLs:

Vendor for Destination Filer	UNIX		NTFS	
	Permission Bits	**UID, GID, time stamps, ...**	**Security Descriptor (SD)**	**Named Streams**
NetApp, UNIX Qtree	copied, then reset *after* file is transferred through CIFS		not copied; derived by destination (NetApp) filer	not supported by SMB/UNIX
NetApp, NTFS Qtree	only high (sticky) bits copied; remaining attributes derived by destination (NetApp) filer		copied *after* UNIX high bits are set	
SMB	copied	copied	not copied	
SMB with CIFS ACLs	copied	copied	not copied; derived by destination (SMB) filer	
EMC	copied, then reset *after* file is transferred through CIFS		copied	

Some CIFS Applications Block Out Migrations

If any client holds a CIFS file open during a file migration, the migration may fail for that file. Applications have the option to universally block *all* CIFS-read access to a file. This is a commonly-used CIFS feature, and it blocks all users including members of the Backup Operator’s group. If an application opens a CIFS file and blocks all access to it, the migration cannot succeed for that file.

You can use the show cifs-service open-files command to get a list of all open, read-locked files (see “Listing Open Files in a CIFS Service” on page 9-9 of the *CLI Maintenance Guide*). To close one from the CLI, use close cifs file (see “Closing an Open File” on page 9-12 of the same manual).

Chapter 13

Grouping Files into Filesets

A *fileset* is a group of files and/or directories to which you can apply replication and migration policies. You can configure filesets based on filename, directory path, size, and/or age. You can create complex filesets by joining multiple filesets in a union or taking the intersection of two or more filesets.

Direct volumes, which contain no metadata, do not support any filesets. This chapter is relevant to managed volumes only.

This chapter explains how to create filesets, and the next chapter explains how to use policies to migrate them to your desired back-end filer(s).

Grouping Files by Filename

You can group files by their names and/or directory path. From gbl mode, use the policy-filename-fileset command to create a name-based fileset:

policy-filename-fileset *name*

where ***name*** (1-64 characters) is a required name that you assign to the fileset.

The CLI prompts for confirmation before creating a new fileset; enter **yes** to continue. This puts you into gbl-ns-vol-fs-name mode, where you can optionally specify paths and/or filenames for files that belong in the set. By default, a new filename fileset matches all files and directories in a volume's root directory, but none below the root.

For example, the following command sequence creates a default named-based fileset, "xmlFiles:"

```
bstnA6k(gbl)# policy-filename-fileset xmlFiles
This will create a new policy object.
```

```
Create object 'xmlFiles'? [yes/no] yes
bstnA6k(gbl-ns-vol-fs-name[xmlFiles])# ...
```

Setting a Directory Path (optional)

You can set a directory path to narrow the scope of the fileset. Only matching files/subdirectories under this path are included in the fileset; the default is the root directory in the managed volume. From gbl-ns-vol-fs-name mode, use the path command to match files in one directory:

path *directory* [ignore-case]

where

directory (1-1024 characters) is a directory in any managed volume where the fileset is used, and

ignore-case (optional) matches the above without considering letter case (for example, **path /docs ignore-case** matches "/docs" and "/Docs").

If a volume that uses this fileset does not contain this path, no files match.

For example, the following command set matches files in /www/xml. If this fileset was used in the wwmed~/acct volume, it would match any files in /acct/www/xml:

```
bstnA6k(gbl)# policy-filename-fileset website
This will create a new policy object.

Create object 'website'? [yes/no] yes
bstnA6k(gbl-ns-vol-fs-name[website])# path /www/xml
bstnA6k(gbl-ns-vol-fs-name[website])# ...
```

Enabling Recursive Matches (optional)

Recursive matches include files from subdirectories. By default, recursive matches are disabled; you can specify them with a wildcard or regular expression (described below), or you can use the recurse command:

recurse

For example, the following command set matches files in /www/xml, including all subdirectories:

```
bstnA6k(gbl)# policy-filename-fileset website
bstnA6k(gbl-ns-vol-fs-name[website])# path /www/xml
bstnA6k(gbl-ns-vol-fs-name[website])# recurse
bstnA6k(gbl-ns-vol-fs-name[website])# ...
```

Disabling Recursive Matches

From gbl-ns-vol-fs-name mode, use the no recurse command to disable recursive matches:

no recurse

For example, the following command set matches files in /log, excluding all subdirectories:

```
bstnA6k(gbl)# policy-filename-fileset logs
bstnA6k(gbl-ns-vol-fs-name[logs])# path /log
bstnA6k(gbl-ns-vol-fs-name[logs])# no recurse
bstnA6k(gbl-ns-vol-fs-name[logs])# ...
```

Matching Against a Wildcard String

To expand your search, you can match against a simple wildcard string to include multiple directories. All matching directories are included in the fileset's scope. To match against wildcards, use the match keyword with a quoted wild-card string:

path match "*wild-card-string*" [ignore-case]

where

wild-card-string (1-1024 characters) uses Unix shell conventions, with one exception (below). The quotes are required. Wild-card conventions are summarized as follows:

- ***** is any string of 0 (zero) or more characters, including an empty string.

- **?** is any single character, or no character.

Note

The * and **?** match any character, including the “/” character. (The “/” is the Unix delimiter between directories.) Therefore, **path match /usr/*/bin** matches both “/usr/local/bin” and “/usr/src/mydir/tmp/bin.” This may be unexpected for Unix users.

- **[...]** matches any one of the enclosed characters. For example, [xyz] matches x, y, or z.
- **[a-z]** matches any character in the sorted range, a through z.
- **[^...]** or **[!...]** matches any character that is *not* enclosed. For example, [!xyz] matches any character *except* x, y, or z.

ignore-case (optional) matches the above without considering letter case.

For example, the following command set matches files in any directory named “xml” or “xsl” (upper or lower-case), such as “/www/xml,” “/www/XSL,” “/var/log/Xml,” “/xsl,” or “/XML:”

```
bstnA6k(gbl)# policy-filename-fileset inXmlOrXslDirs
bstnA6k(gbl-ns-vol-fs-name[inXmlOrXslDirs])# path match "*/x[ms]l" ignore-case
bstnA6k(gbl-ns-vol-fs-name[inXmlOrXslDirs])# ...
```

Using a Complex Regular Expression

You can also match directories using a more complex regular expression. For this option, use the regexp keyword along with a quoted regular expression:

path regexp “*regular-expression*” [ignore-case]

where

regular-expression (1-1024 characters) uses IBM’s ICU conventions for regular expressions (such as “.*\.htm*”, or “[a-z]*\.txt”). The quotes are required. The section below provides details for the regular-expression syntax.

ignore-case (optional) matches the above without considering letter case.

A regular expression makes it possible to be very specific about your matching criteria.

For example, the following command set uses a regular expression to match all hidden Unix directories (directories that start with “/.” and have something other than “.” as their second character):

```
bstnA6k(gbl)# policy-filename-fileset hiddenFiles
bstnA6k(gbl-ns-vol-fs-name[hiddenFiles])# path regexp "/\.[^\.]"
bstnA6k(gbl-ns-vol-fs-name[hiddenFiles])# ...
```

Regular Expression Syntax

The regular expression syntax follows the ICU regular expression conventions, which allow for multiple patterns in the same expression.

Use the following conventions for simple character matches:

. matches any character (including the “/” delimiter between Unix directories).

***** matches 0 (zero) or more of the preceding character or expression. For example, **.*** matches any string, including the null string.

? matches 0 or one of the preceding character or expression. For example, “z?oo” matches either “zoo” or “aloof.”

+ matches one or more of the preceding character or expression.

\ matches the next character, even if that character is a special character. For example, \. matches a period instead of any character, and \? matches a question mark.

Character Groups

[...] matches any one of the enclosed characters. For example, [xyz] matches x, y, or z.

[a-z] matches any character in the sorted range, a through z.

[^...] matches any character that is *not* enclosed. For example, [^xyz] matches any character *except* x, y, or z.

Shorthand for Character Groups

\d matches any numeric digit, 1-9.

\D matches any character except a numeric digit.

\t matches a <Tab>.

\n, **\f**, and **\r** match various flavors of <Enter>. They are <Newline>, <Form Feed>, and <Carriage Return>, respectively.

\s matches any white-space character, [\t\n\f\r\p{Z}]. The “\p{Z}” is any character with the Unicode property, Z.

\S matches any character that is not white space.

Creating Bounds for the Match

{*a*} matches exactly ***a*** of the preceding character or expression. For example, [xyz]{2} matches exactly two characters where each is x, y, or z.

{*a*,*b*} matches at least ***a*** but no more than ***b*** of the preceding character or expression. For example, \d{1,4} matches a number that is 1 to 4 digits long.

^... matches the beginning of a line.

...$ matches the end of a line.

Adding Complexity

(...) defines an atom, and

(*atom1*) | (*atom2*) matches both expressions. For example, “(/var)|(/usr)” matches either “/var/log” or “/usr/local/bin.”

(*atom1* | *atom2*) also matches both expressions.

atom1* | *atom2 also matches both expressions.

Regular-Expression Samples

^/var matches a path with "/var" at its root (for example, "/var/tmp" or "/variable/data," but not "/bin/var").

^/(var | tmp)/ matches two root directories, "/var/" or "/tmp/".

^/home/[^/]+/$ matches any subdirectory of "/home" (such as "/home/juser/") but does not match any directories below that level (such as "/home/juser/misc/").

For More Information

Consult ICU documentation on the Internet for more details on ICU regular expressions.

Negating the Match

There are some cases where you want to match most paths in the volume, with some exceptions. You can use the not keyword in the path command to negate a match, choosing every path that does *not* match the string:

path not *directory* [ignore-case]

matches any directory except the one specified. This only excludes an exact match for *directory.*

path match not "*wild-card-string*" [ignore-case]

matches any directory that does not fit the pattern in the wild-card string.

path regexp not "*regular-expression*" [ignore-case]

matches any directory that does not fit the regular expression.

For example, this command sequence matches all directories in the volume except those named "deleteme:"

```
bstnA6k(gbl)# policy-filename-fileset keepers
bstnA6k(gbl-ns-vol-fs-name[keepers])# path match not "*/deleteme/*"
bstnA6k(gbl-ns-vol-fs-name[keepers])# ...
```

As another example, this command sequence matches all directories except those that start with a "/.":

```
bstnA6k(gbl)# policy-filename-fileset forAllUsers
bstnA6k(gbl-ns-vol-fs-name[forAllUsers])# path regexp not "^/\."
bstnA6k(gbl-ns-vol-fs-name[forAllUsers])# ...
```

Reverting to the Root Path

From gbl-ns-vol-fs-name mode, use the no path command to match files in the volume’s root, “/”:

no path

For example, the following command set configures a name-based fileset, “acctfiles,” to include files in the root:

```
bstnA6k(gbl)# policy-filename-fileset acctFiles
bstnA6k(gbl-ns-vol-fs-name[acctFiles])# no path
bstnA6k(gbl-ns-vol-fs-name[acctfiles])# ...
```

Matching Filenames (optional)

You can use the same methods (above) for specifying the fileset's files. These apply to any files in the chosen path(s). By default, all files match. From gbl-ns-vol-fs-name mode, use the name command to specify certain files for the fileset:

name *filename* [ignore-case]

name match "*wild-card-string*" [ignore-case]

name regexp "*regular-expression*" [ignore-case]

where

filename (1-1024 characters) is an exact filename for the fileset,

match "*wild-card-string*" (1-1024 characters) uses wildcard conventions for shells described above (see "Matching Against a Wildcard String" on page 13-3),

regexp "*regular-expression*" (1-1024 characters) uses IBM's ICU conventions for regular expressions, also described above (see "Regular Expression Syntax" on page 13-5), and

ignore-case (optional) matches the above without considering letter case (for example, **name deleteme ignore-case** matches both "deleteMe" and "DELETEME").

For example, the following command set matches any file with a ".xml" extension. It searches every directory in the volume's tree, recursively:

```
bstnA6k(gbl)# policy-filename-fileset xmlFiles
bstnA6k(gbl-ns-vol-fs-name[xmlFiles])# recurse
bstnA6k(gbl-ns-vol-fs-name[xmlFiles])# name match "*.xml"
bstnA6k(gbl-ns-vol-fs-name[xmlFiles])# ...
```

This next fileset uses a regexp to match any file with a ".fm" or a ".pdf" extension:

```
bstnA6k(gbl)# policy-filename-fileset fm_pdf
bstnA6k(gbl-ns-vol-fs-name[fm_pdf])# name regexp "\.(fm|pdf)$" ignore-case
bstnA6k(gbl-ns-vol-fs-name[fm_pdf])# ...
```

Excluding Files

As with paths, you can use the not keyword to select every file that does *not* match the string:

name not *filename* [ignore-case]

matches any file except the one specified. This only excludes an exact match for *filename*.

name match not "*wild-card-string*" [ignore-case]

excludes any file that fits the pattern in the wild-card string.

name regexp not "*regular-expression*" [ignore-case]

matches any file that does not fit the regular expression.

For example, this command sequence excludes all "*.wmv" and "*.avi" files from the "website" fileset:

```
bstnA6k(gbl)# policy-filename-fileset website
bstnA6k(gbl-ns-vol-fs-name[website])# name regexp not "\.(wmv|avi)$"
bstnA6k(gbl-ns-vol-fs-name[website])# ...
```

Removing the Fileset

Removing a fileset affects file metadata only; it does not delete any files. Use no policy-filename-fileset to remove a name-based fileset:

no policy-filename-fileset *name*

where ***name*** (1-64 characters) identifies the fileset to be removed.

You cannot remove a fileset that is referenced by another fileset or used in a rule. The sections below have configuration instructions for referencing a fileset in these ways.

For example, the following command sequence removes the "httpConf" fileset:

```
bstnA6k(gbl)# no policy-filename-fileset httpConf
bstnA6k(gbl)# ...
```

Grouping Files by Size

You can create filesets based on file size. Each *filesize fileset* contains files "larger-than" or "smaller-than" a size of your choosing, or files in a range between two sizes. From gbl mode, use the policy-filesize-fileset command to create a filesize fileset:

policy-filesize-fileset *name*

where ***name*** (1-64 characters) is the name that you assign to the fileset.

The CLI prompts for confirmation before creating the new fileset; enter **yes** to continue. This puts you into gbl-ns-vol-fs-filesize mode, where you configure the size of files in the fileset.

For example, the following command sequence creates a new filesize fileset:

```
bstnA6k(gbl)# policy-filesize-fileset veryLarge
This will create a new policy object.


Create object 'veryLarge'? [yes/no] yes
bstnA6k(gbl-ns-vol-fs-filesize[veryLarge])# ...
```

Selecting Files Based on their Sizes

The next step in configuring a filesize fileset is to determine a size range for its files. You can select files larger-than (or equal-to) a certain size, smaller-than a certain size, or between two sizes. From gbl-ns-vol-fs-filesize mode, use the select-files command to identify the size for the files:

select-files {larger-than-or-equal-to | smaller-than} *size*[k|M|G|T]

where

larger-than-or-equal-to | **smaller-than** is a required choice,

size is any integer, and

k|**M**|**G**|**T** (optional) sets the size units: **k**ilobytes, **M**egabytes, **G**igabytes, or **T**erabytes. The default is bytes if you omit this. All of these values are 2-based, so a kilobyte is 1024 bytes, a megabytes is 1024*1024, and so on. There can be no spaces between the ***size*** and the unit letter; **20M** is correct, but **20 M** is invalid.

You can run this command twice in the same fileset to choose files between two sizes. For example, you can select-files larger-than-or-equal-to 2M and select-files smaller-than 10M to get a set of files between 2 and 10 Megabytes. You cannot run a select-files command that contradicts a previous one; if you already ran select-files larger-than-or-equal-to 2M, you cannot run select-files smaller-than 1k.

For example, these commands create a fileset with a range of sizes:

```
bstnA6k(gbl)# policy-filesize-fileset veryLarge
bstnA6k(gbl-ns-vol-fs-filesize[veryLarge])# select-files larger-than-or-equal-to 5G
bstnA6k(gbl-ns-vol-fs-filesize[veryLarge])# select-files smaller-than 20G
bstnA6k(gbl-ns-vol-fs-filesize[veryLarge])# ...
```

Removing a File Selection

Use the no select-files command to remove a file selection:

no select-files {larger-than-or-equal-to | smaller-than}

where **larger-than-or-equal-to** | **smaller-than** chooses the selection to remove.

For example, this removes the “smaller-than” selection from the above filesize fileset, making it open-ended:

```
bstnA6k(gbl)# policy-filesize-fileset veryLarge
bstnA6k(gbl-ns-vol-fs-filesize[veryLarge])# no select-files smaller-than
bstnA6k(gbl-ns-vol-fs-filesize[veryLarge])# ...
```

Removing the Fileset

Removing a fileset affects file metadata only; it does not delete any files. From gbl mode, use no policy-filesize-fileset to remove a filesize fileset:

no policy-filesize-fileset *name*

where ***name*** (1-64 characters) identifies the fileset to be removed.

You cannot remove a fileset that is referenced by another fileset or used in a rule. The next chapter has configuration instructions for referencing a fileset from a rule.

For example, the following command sequence removes the “testSize” fileset:

```
bstnA6k(gbl)# no policy-filesize-fileset testSize
bstnA6k(gbl)# ...
```

Grouping Files by Age

You can create filesets based on file age. Each *simple-age fileset* is “older-than” and/or “newer-than” a time of your choosing (6 hours, 3 weeks, or any other time frame). For example, you can create weekly filesets by creating one simple-age fileset that is “newer-than 1 week,” another that is “older-than 1 week” but “newer-than 2 weeks,” and so on.

From gbl mode, use the policy-simple-age-fileset command to create a simple-age fileset:

policy-simple-age-fileset *name*

where ***name*** (1-64 characters) is the name that you assign to the fileset.

The CLI prompts for confirmation before creating the new fileset; enter **yes** to continue. This puts you into gbl-ns-vol-fs-simple-age mode, where you must identify time frame for selecting files (newer-than and/or older-than a particular time). You can also select from several commands that alter the default selection criteria.

For example, the following command sequence creates a new simple-age fileset:

```
bstnA6k(gbl)# policy-simple-age-fileset dayOld
This will create a new policy object.

Create object 'dayOld'? [yes/no] yes
bstnA6k(gbl-ns-vol-fs-simple-age[dayOld])# ...
```

Selecting Files Based on their Ages

The next step in configuring a simple-age fileset is to determine the age for its files. The simple-age fileset defines a single age group for its files: older-than or newer-than a certain time interval. From gbl-ns-vol-fs-simple-age mode, use the select-files command to identify the age group for the files:

> **select-files {older-than | newer-than}** ***count*** **{minutes | hours | days | weeks | months | quarters | years}**
>
> where
>
> **older-than | newer-than** is a required choice.
>
> ***count*** (1-4,294,967,295), and
>
> **minutes | ... years** establish the time frame (for example, **15 minutes**, **2 weeks**, or **1 quarter**).

You can run this command twice in the same fileset to choose files between two ages. For example, you can select-files older-than 2 months and select-files newer-than 3 months to get a set of files between 2 and 3 months old. You cannot run a select-files command that contradicts a previous one; if you already ran select-files newer-than 1 week, you cannot run select-files older-than 2 quarters.

For example, the following command set selects files for the "dayOld" fileset:

```
bstnA6k(gbl)# policy-simple-age-fileset dayOld
bstnA6k(gbl-ns-vol-fs-simple-age[dayOld])# select-files older-than 1 days
bstnA6k(gbl-ns-vol-fs-simple-age[dayOld])# ...
```

Removing a File Selection

Use the no select-files command to remove a file selection:

no select-files {older-than | newer-than}

where **older-than | newer-than** chooses the selection to remove.

For example, this removes the “older-than” selection from a simple-age fileset:

```
bstnA6k(gbl)# policy-simple-age-fileset 2mo
This will create a new policy object.

Create object '2mo'? [yes/no] yes
bstnA6k(gbl-ns-vol-fs-simple-age[2mo])# no select-files older-than
bstnA6k(gbl-ns-vol-fs-simple-age[2mo])# ...
```

Choosing Last-Accessed or Last-Modified

The next step in configuring a simple-age fileset is to determine whether it selects files by last-accessed time or last-modified time. The default is last-modified time. From gbl-ns-vol-fs-simple-age mode, use the last command to choose the age type:

last {accessed | modified}

where **accessed | modified** is a required choice.

Note

We do not recommend using last-accessed times for selecting directories in CIFS or multi-protocol (CIFS and NFS) namespaces. CIFS filers update a directory’s last-accessed time whenever the policy engine reads the time stamp; this can cause unexpected directory migrations for directories that have not been accessed by any clients. NFS-only filers do not have this problem, nor is it a problem with file access.

For example, the following command set configures the “dayOld” fileset to select its files based on last-accessed time:

```
bstnA6k(gbl)# policy-simple-age-fileset dayOld
bstnA6k(gbl-ns-vol-fs-simple-age[dayOld])# last accessed
bstnA6k(gbl-ns-vol-fs-simple-age[dayOld])# ...
```

Identifying a Source Fileset (optional)

By default, the fileset selects its files from all of files in the current volume. You can narrow this scope by choosing a source fileset (for example, a filename fileset with a particular directory path). If this is set, the select-files command chooses from the pool of files in the source fileset. From gbl-ns-vol-fs-simple-age mode, use the from fileset command to identify a source fileset:

from fileset *fileset-name*

where ***fileset-name*** (1-64 characters) is the source fileset. Use show policy filesets (described later in this chapter) for a list of global filesets, or show policy *namespace volume* for a list of filesets in the current volume.

For example, the following command set uses the “website” fileset as the source for the “2mo” simple-age fileset:

```
bstnA6k(gbl)# policy-simple-age-fileset 2mo
bstnA6k(gbl-ns-vol-fs-simple-age[2mo])# from fileset website
bstnA6k(gbl-ns-vol-fs-simple-age[2mo])# ...
```

Removing the Source Fileset

As mentioned above, a source fileset is not required for a simple-age fileset. Without the source fileset, the fileset selects from all files in the volume. Use no from to remove the source fileset:

no from

For example, the following command set removes the source fileset from from the “dayOld” fileset:

```
bstnA6k(gbl)# policy-simple-age-fileset dayOld
bstnA6k(gbl-ns-vol-fs-simple-age[dayOld])# no from
bstnA6k(gbl-ns-vol-fs-simple-age[dayOld])# ...
```

Setting the Age-Evaluation Interval (optional)

By default, the simple-age fileset selects all of its files whenever it is used by a rule. For example, suppose a file-placement rule uses a simple-age fileset that selects files newer than 3 hours. (A later chapter explains how to use a file-placement rule with filesets.) Every time the file-placement rule runs, the simple-age fileset selects the files that are newer than three hours *at that moment*. If this is sufficient, you can skip this section.

Some installations may want to regulate the file selection. You can configure a simple-age fileset to select its files on a schedule, such as "every 2 hours" or "every Wednesday." Whenever any rule uses a scheduled fileset, the files are already selected. From gbl-ns-vol-fs-simple-age mode, use the every command to choose a regular age-evaluation interval for the current fileset:

every *number* {minutes | hours | days | weeks | months | quarters | years}

or

every *day-of-week*

where you can choose between two formats:

- **every *number unit***, where the *number* is an integer and *unit* is minutes, hours, days, weeks, months, or years. For example, **every 15 minutes**, **every 3 weeks**, **every 4 days**, or **every 1 years**.
- **every *day-of-week***, where the *day-of-week* is the full name of the day in lower case. For example, **every monday**, **every wednesday**, or **every sunday**.

For example, the following command set configures the "dayOld" fileset to re-select its files every half hour:

```
bstnA6k(gbl)# policy-simple-age-fileset dayOld
bstnA6k(gbl-ns-vol-fs-simple-age[dayOld])# every 30 minutes
bstnA6k(gbl-ns-vol-fs-simple-age[dayOld])# ...
```

Reverting to the Default Evaluation Interval

By default, a simple-age fileset selects its files whenever it is used by a rule. To return to this default, use the no every command from gbl-ns-vol-fs-simple-age mode:

no every

For example, the following command set configures the “dayOld” fileset to select its files on demand:

```
bstnA6k(gbl)# policy-simple-age-fileset dayOld
bstnA6k(gbl-ns-vol-fs-simple-age[dayOld])# no every
bstnA6k(gbl-ns-vol-fs-simple-age[dayOld])# ...
```

Setting the File-Selection Start Time (optional)

The last step in configuring a simple-age fileset is to set a start time. This is optional; like the every command, it creates a regular schedule for selecting files. If used without the every command, the fileset selects its files weekly.

The start time marks the time-of-day and/or day-of-the-week to start its selection schedule. For example, if you choose 1 PM today for a fileset that selects files every 2 days, the policy engine selects files today at 1PM, two days from now at 1PM, four days from now at 1PM, and so on. You can also use this command to postpone all file selections until some future date and time. From gbl-ns-vol-fs-simple-age mode, use the start command to choose the start date and time:

start *date*:*time*

where the colon (**:**) separates the date from the time.

date defines the start date for the fileset. Use the format ***mm*/*dd*/*yyyy*** (for example, **04/01/2003** for April 1, 2003, or **11/20/2005** for November 20, 2005).

time defines the start time. Use the format ***HH*:*MM*:*SS*** (for example, **12:00:00** for noon, or **17:00:00** for 5 PM).

For example, the following command set configures the “dayOld” fileset to start its file-selection schedule at 4 AM on May 6, 2005:

```
bstnA6k(gbl)# policy-simple-age-fileset dayOld
bstnA6k(gbl-ns-vol-fs-simple-age[dayOld])# start 05/06/2005:04:00:00
```

```
bstnA6k(gbl-ns-vol-fs-simple-age[dayOld])# ...
```

Reverting to the Default Start Time

By default, a simple-age fileset selects its files whenever it is used by a rule. If the fileset uses a schedule set by the every command, the default start time is the time that an administrator entered the every command. To return to the default, use the no start command from gbl-ns-vol-fs-simple-age mode:

no start

For example:

```
bstnA6k(gbl)# policy-simple-age-fileset dayOld
bstnA6k(gbl-ns-vol-fs-simple-age[dayOld])# no start
bstnA6k(gbl-ns-vol-fs-simple-age[dayOld])# ...
```

Removing the Fileset

Removing a fileset affects file metadata only; it does not delete any files. From gbl mode, use no policy-simple-age-fileset to remove a simple-age fileset:

no policy-simple-age-fileset *name*

where ***name*** (1-64 characters) identifies the fileset to be removed.

You cannot remove a fileset that is referenced by another fileset or used in a rule. The next chapter has configuration instructions for referencing a fileset from a rule.

For example, the following command sequence removes the “testSimpleAge” fileset:

```
bstnA6k(gbl)# no policy-simple-age-fileset testSimpleAge
bstnA6k(gbl)# ...
```

Joining Filesets

You can join two or more filesets in a union fileset. A *union* fileset contains all the files in all of its source filesets. A file that is common to two or more of the source filesets is only included once in the resulting union. From gbl mode, use the policy-union-fileset command to create a union fileset:

policy-union-fileset *name*

where ***name*** (1-64 characters) is a required name that you assign to the fileset.

The CLI prompts for confirmation before creating a new fileset; enter **yes** to continue. This puts you into gbl-ns-vol-fs-union mode, where you must identify two or more source filesets to include in the union.

For example, the following command sequence creates an empty union fileset:

```
bstnA6k(gbl)# policy-union-fileset bulky
This will create a new policy object.

Create object 'bulky'? [yes/no] yes
bstnA6k(gbl-ns-vol-fs-union[bulky])# ...
```

Identifying a Source Fileset

The final step in configuring a union fileset is to identify two or more source filesets. You can include as many source filesets as desired, but you must have at least two for the union to be meaningful. From gbl-ns-vol-fs-union mode, use the from fileset command to include a source fileset:

from fileset *fileset-name*

where ***fileset-name*** (1-64 characters) identifies the source fileset.

For example, the following command set includes three source filesets for the “bulky” union:

```
bstnA6k(gbl)# policy-union-fileset bulky
bstnA6k(gbl-ns-vol-fs-union[bulky])# from fileset fm_pdf
bstnA6k(gbl-ns-vol-fs-union[bulky])# from fileset veryLarge
```

```
bstnA6k(gbl-ns-vol-fs-union[bulky])# from fileset xmlFiles
bstnA6k(gbl-ns-vol-fs-union[bulky])# ...
```

Removing a Source Fileset

Use the no form of the from fileset command to remove a fileset from the list of sources:

no from fileset *fileset-name*

where ***fileset-name*** (1-64 characters) identifies the fileset to remove.

You cannot remove the last source fileset if the union fileset is in use (that is, referenced by another fileset or used in a rule). The next chapter has configuration instructions for referencing a fileset from a rule.

For example, the following command set removes a source fileset from the "bulky" fileset:

```
bstnA6k(gbl)# policy-union-fileset bulky
bstnA6k(gbl-ns-vol-fs-union[bulky])# no from fileset xmlFiles
bstnA6k(gbl-ns-vol-fs-union[bulky])# ...
```

Removing All Source Filesets

Use no from all to remove all source filesets at once:

no from all

As mentioned above, you cannot remove all source filesets if the union fileset is in use.

For example:

```
bstnA6k(gbl)# policy-union-fileset testUnion
bstnA6k(gbl-ns-vol-fs-union[testUnion])# no from all
bstnA6k(gbl-ns-vol-fs-union[testUnion])# ...
```

Removing the Fileset

Removing a fileset affects the switch configuration only; it does not delete any files. Use no policy-union-fileset to remove a union fileset from the current volume:

no policy-union-fileset *name*

where ***name*** (1-64 characters) identifies the fileset to be removed.

You cannot remove a fileset that is referenced by another fileset or used in a rule. The next chapter has configuration instructions for referencing a fileset from a rule.

For example, the following command sequence removes the “allCustomerSide” fileset:

```
bstnA6k(gbl)# no policy-union-fileset allCustomerSide
bstnA6k(gbl)# ...
```

Intersecting Filesets

You can intersect two or more filesets into an intersection fileset. An *intersection* fileset contains files that are common to all of its source filesets; a file must be in all of the source filesets to be included in the intersection. From gbl mode, use the policy-intersection-fileset command to create an intersection fileset:

policy-intersection-fileset *name*

where ***name*** (1-64 characters) is a required name that you assign to the fileset.

As with any fileset, the CLI prompts for confirmation before creating a new intersection fileset; enter **yes** to continue. This puts you into gbl-ns-vol-fs-isect mode, where you identify two or more source filesets to intersect.

For example, the following command sequence creates an empty intersection fileset in the “/acct” volume:

```
bstnA6k(gbl)# policy-intersection-fileset paidBills
This will create a new policy object.

Create object 'paidBills'? [yes/no] yes
bstnA6k(gbl-ns-vol-fs-isect[paidBills])# ...
```

Identifying a Source Fileset

The final step in configuring an intersection fileset is to identify two or more source filesets. You can include as many source filesets as desired, but you must have at least two for the intersection to be meaningful. From gbl-ns-vol-fs-isect mode, use the from fileset command to include a source fileset:

from fileset *fileset-name*

where ***fileset-name*** identifies the source fileset.

For example, the following command set intersects two source filesets to make the “paidBills” fileset:

```
bstnA6k(gbl)# policy-intersection-fileset paidBills
bstnA6k(gbl-ns-vol-fs-isect[paidBills])# from fileset webdavPaid
bstnA6k(gbl-ns-vol-fs-isect[paidBills])# from fileset wdValid
bstnA6k(gbl-ns-vol-fs-isect[paidBills])# ...
```

Removing a Source Fileset

Use the no form of the from fileset command to remove a fileset from the list of sources:

no from fileset *fileset-name*

where ***fileset-name*** identifies the fileset to remove.

You cannot remove the last source fileset if the intersection fileset is in use (that is, referenced by another fileset or used in a rule). The next chapter has configuration instructions for referencing a fileset from a rule.

For example, the following command set removes two source filesets from the “paidBills” fileset:

```
bstnA6k(gbl)# policy-intersection-fileset paidBills
bstnA6k(gbl-ns-vol-fs-isect[paidBills])# no from fileset webdavUnderwritten
bstnA6k(gbl-ns-vol-fs-isect[paidBills])# no from fileset monthlyPaid group 2
bstnA6k(gbl-ns-vol-fs-isect[paidBills])# ...
```

Removing All Source Filesets

To remove all filesets with a single command, use no from all:

no from all

As above, you cannot remove all source filesets if the intersection fileset is in use.

For example, the following command set removes all source filesets from the "paidBills" fileset:

```
bstnA6k(gbl)# policy-intersection-fileset paidBills
bstnA6k(gbl-ns-vol-fs-isect[paidBills])# no from all
bstnA6k(gbl-ns-vol-fs-isect[paidBills])# ...
```

Removing the Fileset

Removing a fileset affects file metadata only; it does not delete any files. Use no policy-intersection-fileset to remove an intersection fileset:

no policy-intersection-fileset *name*

where ***name*** (1-64 characters) identifies the fileset to be removed.

You cannot remove a fileset that is referenced by another fileset or used in a rule. The next chapter has configuration instructions for referencing a fileset from a rule.

For example, the following command sequence removes the "paidBills" fileset:

```
bstnA6k(gbl)# no policy-intersection-fileset paidBills
bstnA6k(gbl)# ...
```

Listing all Filesets

Use the show policy filesets command to show all filesets:

show policy filesets

This command shows the configuration for all global filesets (configured in gbl mode), followed by all filesets configured in each managed volume.

For example:

```
bstnA6k(gbl)# show policy filesets

Global Policy:

  Filename Fileset:  website

    Configuration:
      Name Does Not Match Regular Expression:    \.(wmv|avi)$
      Path Is:                                   /www/xml/
      Recurse:                                   Yes


  Filename Fileset:  hiddenFiles

    Configuration:
      Name Is:
      Path Matches Regular Expression:           /\.[^\.]
      Recurse:                                   No


  Filename Fileset:  fm_pdf

    Configuration:
      Name Matches Regular Expression:           \.(fm|pdf)$ (case ignored)
      Recurse:                                   Yes


  Filename Fileset:  xmlFiles

    Configuration:
      Name Matches Pattern:                      *.xml
      Recurse:                                   Yes


  File Size Fileset:  veryLarge
```

```
    Configuration:
      Select Files Larger Than Or Equal To:      5.0G


  Fileset Union:    bulky

    Configuration:
      From fileset:                              fm_pdf
      From fileset:                              veryLarge


  Simple Age Fileset:  dayOld

    Configuration:
      Select files older than:                   1 days
      Mode:                                      Last Accessed


  Simple Age Fileset:  2mo

    Configuration:
      From fileset:                              website
      Select files newer than:                   2 months
      Mode:                                      Last Modified

Namespace:        medco
Namespace:        wwmed
Volume:           /acct
Namespace:        medarcv
Volume:           /rcrds
Volume:           /lab_equipment
Namespace:        insur
Volume:           /claims
```

```
  Filename Fileset:  images

    Configuration:
      Name Is:
      Path Is:                                         /images/
      Recurse:                                         Yes

bstnA6k(gbl)# ...
```

Showing One Global Fileset

To show a single global fileset, add the global-fileset argument to the end of the show policy filesets command:

show policy filesets global-fileset *fileset-name*

where ***fileset-name*** (optional, 1-1024 characters) chooses the fileset.

For example, the following command shows the “fm_pdf” fileset:

```
bstnA6k(gbl)# show policy filesets global-fileset fm_pdf

Global Policy:

  Filename Fileset:  fm_pdf

    Configuration:
      Name Matches Regular Expression:                 \.(fm|pdf)$ (case ignored)
      Recurse:                                         Yes

bstnA6k(gbl)# ...
```

Showing Filesets in a Managed Volume

All filesets can be alternatively configured within a managed volume. To show the configuration for such a fileset, specify the namespace and volume name before the fileset name:

show policy filesets *namespace volume fileset-name*

where

namespace (1-30 characters) is the fileset's namespace.

volume (optional, 1-1024 characters) identifies the fileset's volume.

fileset-name (optional, 1-1024 characters) chooses the fileset.

Sample - Configuring Age-Based Filesets

The following command sequence creates several filesets as building blocks, then joins them together in a union fileset. This creates a fileset out of *.xls and *.doc files in /xyz/public that have been accessed in the last 30 days:

First, create a fileset that recursively matches all *.xls files in /xyz/public:

```
bstnA6k(gbl)# policy-filename-fileset xls_files
bstnA6k(gbl-ns-vol-fs-name[xls_files])# name match "*.xls"
bstnA6k(gbl-ns-vol-fs-name[xls_files])# path /xyz/public
bstnA6k(gbl-ns-vol-fs-name[xls_files])# recurse
bstnA6k(gbl-ns-vol-fs-name[xls_files])# exit
```

Create a similar fileset for *.doc and *.pdf files:

```
bstnA6k(gbl)# policy-filename-fileset doc_files
bstnA6k(gbl-ns-vol-fs-name[doc_files])# name regexp "\.(doc|pdf)$"
bstnA6k(gbl-ns-vol-fs-name[doc_files])# path /xyz/public
bstnA6k(gbl-ns-vol-fs-name[doc_files])# recurse
bstnA6k(gbl-ns-vol-fs-name[doc_files])# exit
```

Join the two filesets in a union:

```
bstnA6k(gbl)# policy-union-fileset office_files
This will create a new policy object.
```

```
Create object 'office_files'? [yes/no] yes
bstnA6k(gbl-ns-vol-fs-union[office_files])# from fileset xls_files
bstnA6k(gbl-ns-vol-fs-union[office_files])# from fileset doc_files
bstnA6k(gbl-ns-vol-fs-union[office_files])# exit
```

In the “wwmed~/acct” volume, create an age-based fileset that only takes the office files that were accessed this month:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# simple-age-fileset thisMonth
This will create a new policy object.

Create object 'thisMonth'? [yes/no] yes
bstnA6k(gbl-ns-vol-fs-simple-age[thisMonth])# select-files newer-than 1 months
bstnA6k(gbl-ns-vol-fs-simple-age[thisMonth])# last accessed
bstnA6k(gbl-ns-vol-fs-simple-age[thisMonth])# from fileset office_files
bstnA6k(gbl-ns-vol-fs-simple-age[thisMonth])# end
bstnA6k#
```

Chapter 14

Migrating Filesets

A *fileset* is a group of files and/or directories to which you can apply replication and migration policies. This chapter explains how to configure policies for migrating filesets to desired back-end storage.

Direct volumes, which contain no metadata, do not support any filesets. This chapter is relevant to managed volumes only.

Before You Begin

You must create one or more filesets for the fileset policy. See Chapter 13, *Grouping Files into Filesets*.

Concepts and Terminology

When files migrate from one share to another, their parent directories are duplicated on the destination share. The directories are said to be *striped* across the two shares. The original directory, the one that was first-imported, is called the *master directory.* A new file or directory goes to its parent's master directory by default, so that directory trees tend to grow on a single back-end share.

Directing File Placement

You can use fileset policies to steer files and/or directories onto specific storage targets. You choose the files/directories by configuring a fileset for them, and you choose the storage target by creating a *placement rule* in the volume. This is the same placement rule that can move everything off of a share (refer back to "Draining One or More Shares" on page 12-35); the difference is that you can use a dynamic fileset as the source rather than a static share. The target for file placement can be a share or a share farm in the current volume.

For example, you could direct all of the volume's HTML files to a particular share, or you could migrate an entire directory tree from one share to another. You could also configure the switch to move all files that have not been read for over a week onto a share farm with slower back-end performance.

From gbl-ns-vol mode, use place-rule to create a new placement rule:

place-rule *name*

where ***name*** (1-64 characters) is a name you choose for the rule.

This puts you into gbl-ns-vol-plc mode, where you must choose a source fileset, set the storage target for the files, and enable the rule. There are also some optional commands you can invoke from this mode.

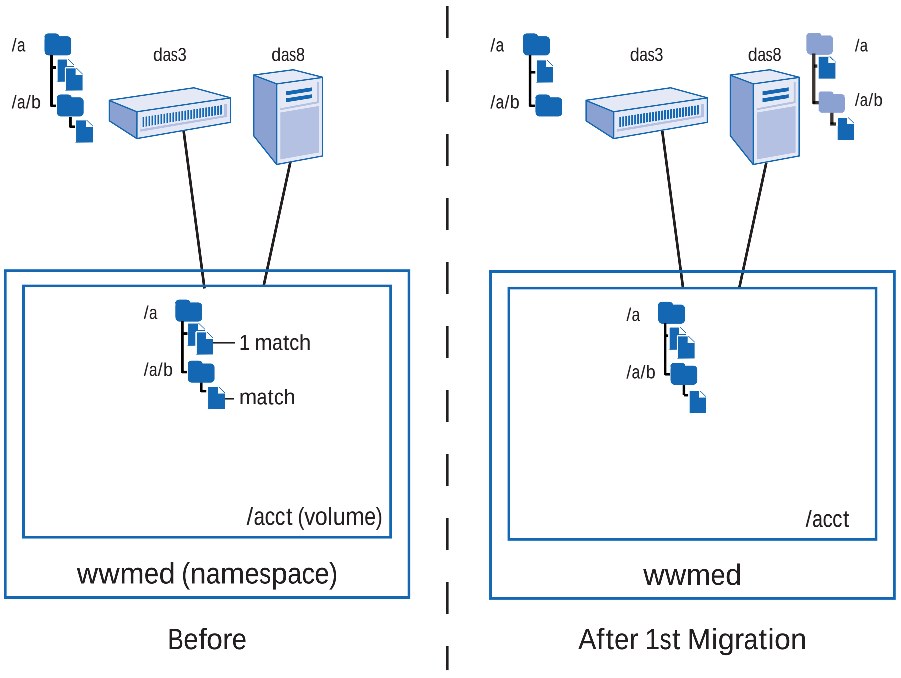
For example, the following command sequence creates an empty placement rule, "docs2das8," for the volume, "wwmed~/acct:"

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule docs2das8
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# ...
```

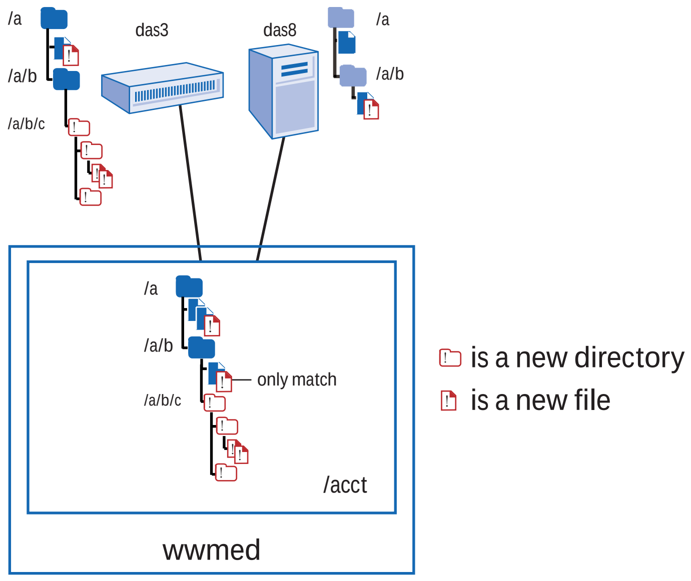
Identifying the Source Fileset

The next step in configuring a placement rule is to identify the source fileset. This chooses a set of files and/or directories based on their names, sizes, ages, or other criteria; this set of files and/or directories changes as clients create, edit, and delete them in the volume. The placement rule migrates all selections from the source fileset onto the target storage, then it steers any new matching files/directories as clients create and edit them.

You can make the placement rule select files only, directories only, or both; by default, the rule selects files only. The default applies the fileset exclusively to filenames. Directories are copied (or *striped*) only as needed to hold the placed files. For example, the illustration below shows a small directory tree on das3 containing two matching files. The directories that contain the matching files, /a and /a/b, are striped to das8 so that das8 can hold those files. One file in directory /a does not match, so it remains in the master directory on das3.



Note that the master copies of the directories remain on their original filer, das3; this means that new files in those directories go to das3 by default, if they are outside the fileset. All of their new subdirectories go to das3, too. In this illustration, only one new file matches and is steered onto das8. All new directories and non-matching files are created on das3, which has all of the master directories. Had the rule been removed after the initial migration, above, *all* of the new files would have followed their master directories onto das3.



From gbl-ns-vol-plc mode, use the from fileset ... match files command to apply a source fileset to files only. This results in the behavior illustrated above:

from fileset *fileset-name* [match files]

where

fileset-name (1-64 characters) identifies the source fileset, and

match files (optional) applies the fileset only to files, not directories. This is the default, so it is optional.

For example, the following command set selects files that match the “fm_pdf” fileset:

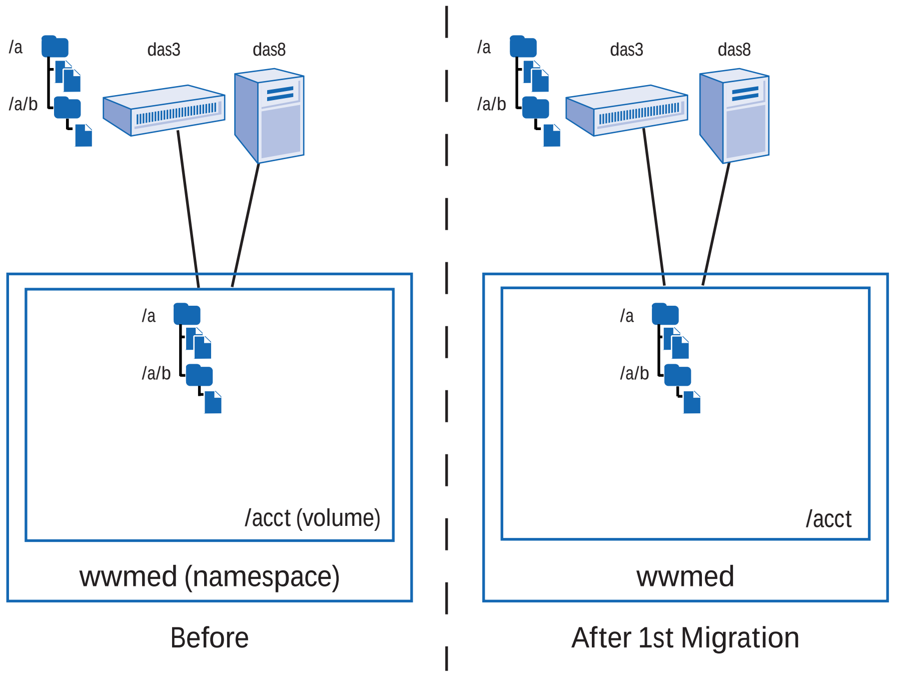
```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
```

```
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule docs2das8
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# from fileset fm_pdf match files
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# ...
```

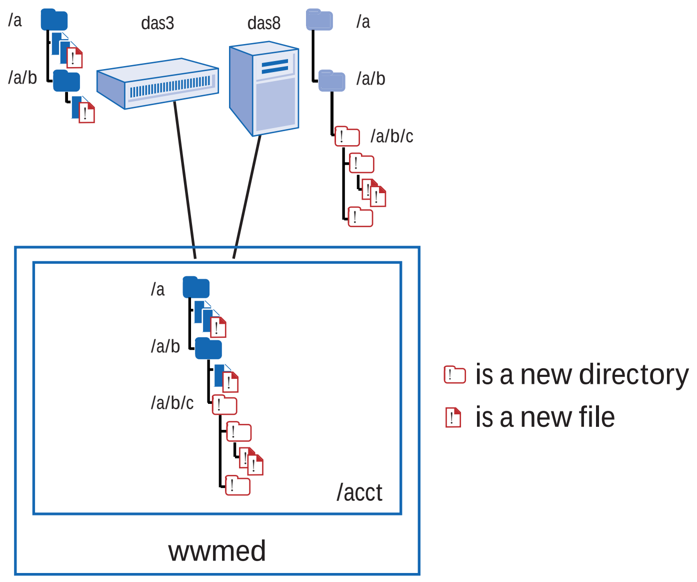
Matching Directories Only

Consider a directory tree on a filer that is filled to a comfortable level, but clients are likely to add subdirectories that may overfill the share. You want all new subdirectories to be created on a new filer and grow there, but existing files and directories can stay. Also, new files in the existing directories can stay on the original filer.

You can use the from fileset command to select directories only, so that the fileset's criteria is not applied to any files. This steers new directories to das8. Old directories keep their masters on das3, but new directories are created at das8 and therefore have their masters there. Files are not matched, so none of the existing files migrate. The initial run of the rule does nothing:



This configuration mainly focuses on new directories and their sub-trees. The placement rule steers new directories to das8. Since the new directories are created on das8, the das8 instance of the directory is master. By default, all of its child files and directories follow it onto das8. Directory /a/b/c therefore grows on das8, as would any other new directories in the volume. Files are not matched by this rule, so they always go to the filer that holds their parent's master directory; the old directories (/a and /a/b) therefore store all new files on das3, and files in the new directories go to das8. This configuration is designed to move all of the major tree growth onto the target filer.



In the from fileset command, you can use the match directories argument to choose directories only. This causes the placement illustrated above:

from fileset *fileset-name* match directories

where

fileset-name (1-64 characters) identifies the source fileset, and

match directories applies the fileset only to directories, not files.

For example, the following command set makes an all-inclusive fileset and uses it to match all directories:

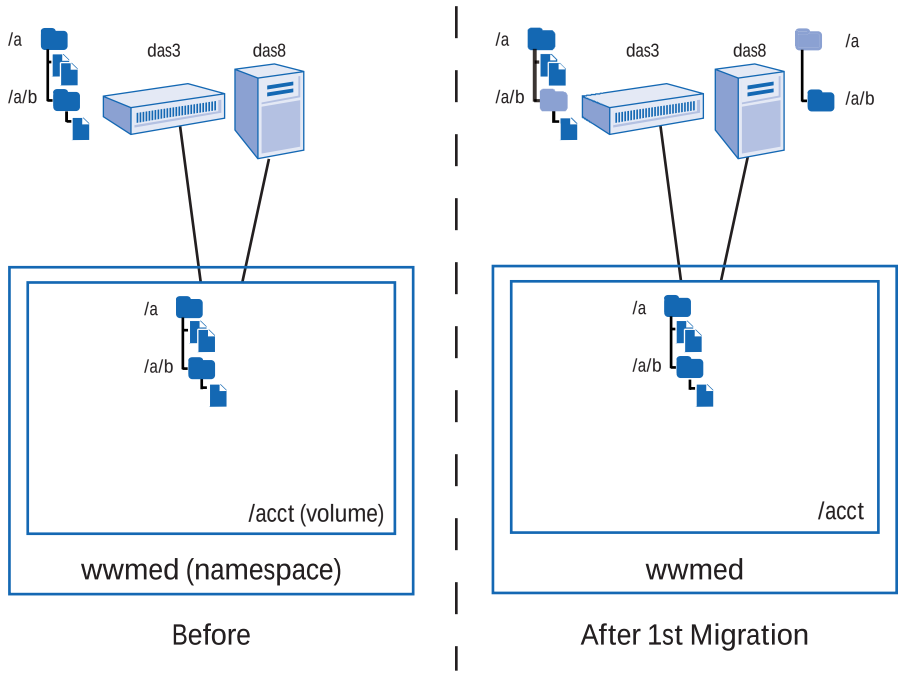
```
bstnA6k(gbl)# policy-filename-fileset all
bstnA6k(gbl-ns-vol-fs-name[all])# recurse
bstnA6k(gbl-ns-vol-fs-name[all])# exit
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule noNewDirs
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~noNewDirs])# from fileset all match directories
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~noNewDirs])# ...
```

Matching and Promoting Directories

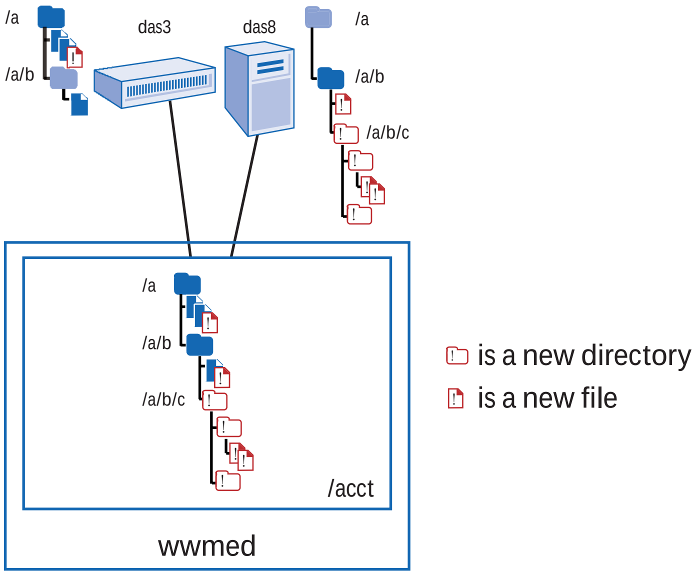
Both of the previous from fileset commands did not change the current master directories; das3 kept the master versions of both /a and /a/b. You may want to promote the migrated directories to master, so that the source filer keeps all existing files but the target filer gets all of their *new* files and subdirectories by default. This can be useful after rebuilding a managed volume with several filers: a directory tree that was previously constrained to das8 may be re-imported with its masters on other filers. By promoting all matching directories to master, you can return sole mastership to das8. This causes the directories to grow on das8, not on various filers throughout the volume.

(For instructions on rebuilding a volume, refer to "Rebuilding a Volume" on page 5-35 of the *CLI Maintenance Guide*.)

For example, suppose das8 is supposed to be master of /a/b. You can promote it and all of its descendant directories without migrating any files. After the placement rule runs, the file already under /a/b stays on das3. The master for /a/b is now on das8. The /a directory is striped to das8 so that it can hold the /a/b directory; its master remains on das3.



As clients add new files and subdirectories to /a/b, they go onto das8 instead of das3. New files in /a, which is outside the fileset, continue to gravitate to das3.



In the from fileset command, you can add the promote-directories flag to promote the chosen directories. This causes the placement illustrated above:

from fileset *fileset-name* match directories promote-directories

where

fileset-name (1-64 characters) identifies the source fileset,

match directories applies the fileset only to directories, not files, and

promote-directories promotes all matching directories on the target filer to master.

For example, the following command set creates a fileset to match the /a/b tree, matches the fileset against directories, and promotes the matching directories:

```
bstnA6k(gbl)# policy-filename-fileset a_b_tree
bstnA6k(gbl-ns-vol-fs-name[a_b_tree])# path match /a/b
bstnA6k(gbl-ns-vol-fs-name[a_b_tree])# recurse
bstnA6k(gbl-ns-vol-fs-name[a_b_tree])# exit
bstnA6k(gbl)# namespace wwmed
```

```
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule reset
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~reset])# from fileset a_b_tree match directories
promote-directories
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~reset])# ...
```

Promoting Directories on a Target Share Farm

If the file-placement target is a share farm, the share that gets the directory also gets the directory promotion. The share farm's new-file-placement algorithm determines the exact share where the directory goes. Share-farm rules for new-file placement were described earlier: recall "Balancing New Files Based on Free Space" on page 12-19.

Avoid Promoting CIFS Directories Based on Last-Accessed Time

CIFS filers update a directory's last-accessed time whenever the policy engine reads the time stamp. Do not migrate and promote CIFS directories if the source fileset

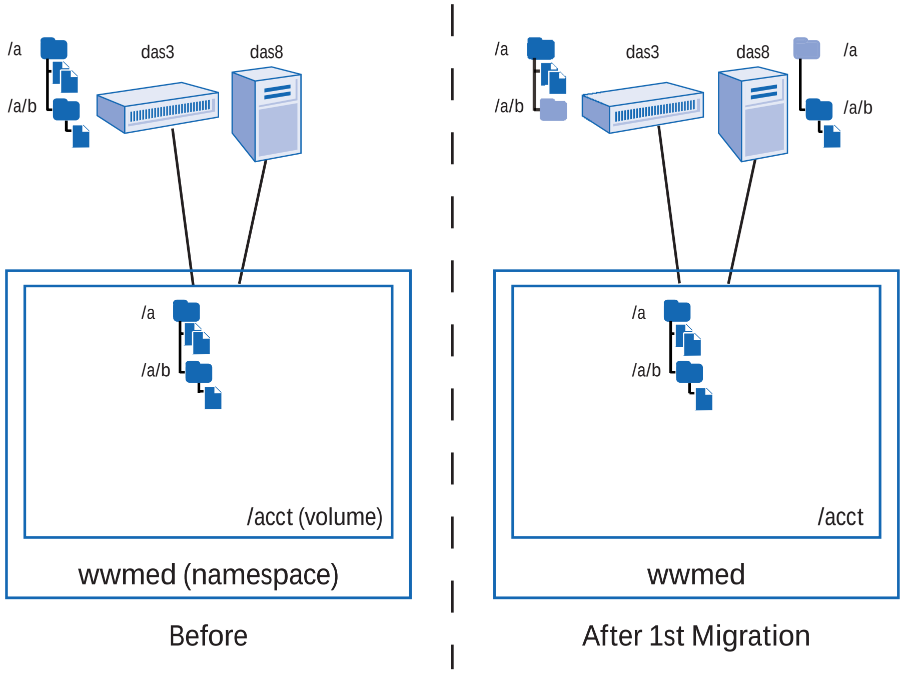
- is age-based, and
- does all selections based in the last-accessed time (recall "Choosing Last-Accessed or Last-Modified" on page 13-15).

If a rule migrates and promotes directories based on their last-accessed times in a CIFS or multi-protocol namespace, the back-end filer changes the directory's time stamp after the migration. Any other rule that chooses directories based on the last-accessed time would therefore use the wrong time stamp. This can cause unpredictable results.

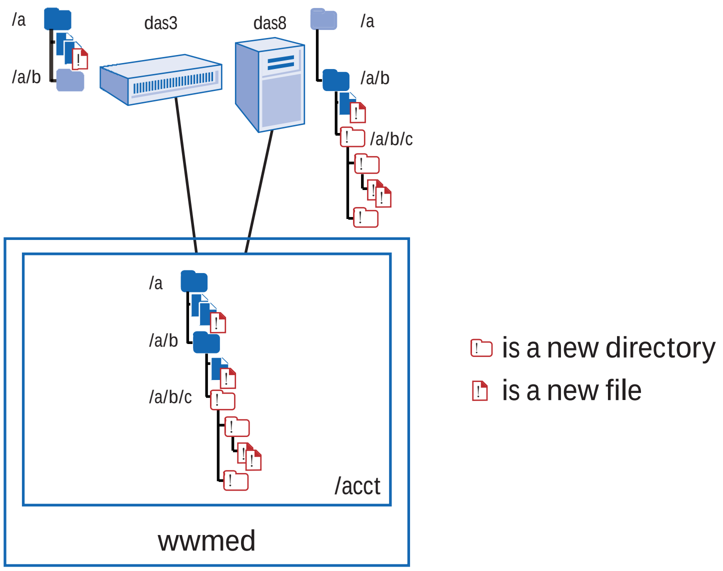
NFS-only filers do not have this problem, nor is it a problem with files.

Matching Directory Trees (Directories and Files)

By combining files, directories, and directory promotion, you can move an entire directory tree from das3 to das8 and make it grow on das8. For example, you can migrate all existing files in directory /a/b in addition to ensuring that new files and directories get created on das8. The only difference between this and the previous example is that the existing file in /a/b also migrates over to das8:



Each new file and directory, as in the previous example, follows its parent's master directory. Directory /a/b grows on das8 while new files in /a continue to gravitate to das3:



In the from fileset command, you can use match all to match both files and directories. Keep the promote-directories flag, to make the selected directories (such as /a/b/c) "master" at the target filer. This causes the placement illustrated above:

from fileset *fileset-name* match all promote-directories

where

fileset-name (1-64 characters) identifies the source fileset,

match all applies the fileset to both files and directories, and

promote-directories promotes all matching directories on the target filer to master.

For example, the following command set migrates the directories that match the "a_b_tree" fileset:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule mvtree
```

```
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~mvtree])# from fileset a_b_tree match all
promote-directories
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~mvtree])# ...
```

Limiting the Selection to Particular Source Share(s) (optional)

You can select the above files and/or directories from a particular share or share farm. By default, the selected files and directories come from all shares in the volume. From gbl-ns-vol-plc mode, use one of two source commands to specify the source share(s):

source share *share-name*

where **share *share-name*** (1-64 characters) identifies a single source share, or

source share-farm *share-farm-name*

where ***share-farm-name*** (1-64 characters) is a group of shares in a share farm.

Use the show global-config namespace command to see the shares or share farms in each volume: see "Showing Namespace Configuration" on page 7-25.

For example, the following command set limits the "mvtree" rule to migrate off of the "bills2" share only:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule mvtree
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~mvtree])# source share bills2
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~mvtree])# ...
```

Removing all Source-Share Restrictions

Use the no source command to remove any restrictions on source shares. This causes the placement rule to select files and/or directories from all shares in the current volume:

no source

For example, the following command set allows the "mvtree" rule to select its files and directories from all shares:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule mvtree
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~mvtree])# no source
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~mvtree])# ...
```

Choosing the Target Storage

You choose the target storage for a fileset with the same commands that are used for emptying a share.Either a share or a share farm is a valid target for the fileset. From gbl-ns-vol-plc mode, use one of two target rules to set the fileset's storage target:

target share *share-name*

where ***share-name*** (1-64 characters) is a share from the current volume. Use the show global-config namespace command to see the shares in each volume: see "Showing Namespace Configuration" on page 7-25.

target share-farm *share-farm-name*

where ***share-farm-name*** (1-64 characters) is a share farm within the current volume. The show namespace command also shows the share farms in each of the namespace's volumes.

For example, the following command sequence selects a share, "bills," as the home for all files matched by the "docs2das8" rule. (The volume share, "bills," maps to a share on the "das8" filer.)

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule docs2das8
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# target share bills
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# ...
```

As another example, the following command sequence selects a share farm, "fm3," as the target for a different placement rule:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule distributeFiles
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~distributeFiles])# target share-farm fm3
```

```
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~distributeFiles])# ...
```

Balancing Capacity in a Target Share Farm

When a share farm is a file-placement target, the first configured share in the farm is the default share for placed files. Most files are placed on the same share as their parent directory, but a file defaults to the first share if its parent directory is outside the share farm. The first share in the farm can therefore take a heavier file burden over time.

To migrate files off of any share that is running low on free space, you can configure *auto migration* for the share farm. Refer back to "Auto Migrating Existing Files" on page 12-18 to configure auto migration.

Limiting Each Migration (optional)

You can use the limit-migrate command to put a ceiling on the amount of data migrated. The policy engine migrates files until it meets this limit; it stops migrating as soon as it discovers that the next file would exceed the limit.

limit-migrate *size*[k|m|g|t]

where

size (1-18,446,744,073,709,551,615) is the size, and

k|**m**|**g**|**t** (optional) is the units; **k**ilobytes (1024 bytes), **m**egabytes (1024*1024 bytes), **g**igabytes (1024*1024*1024), or **t**erabytes (1024*1024*1024*1024). The default is bytes.

This limit applies to every run of the placement rule, so you can use it in conjunction with a schedule to migrate a limited amount of data during off-hours. For example, you can allow a limit of 10 Gigabytes and a daily schedule that runs at midnight. If the source fileset contains 50 Gigabytes of data, it would migrate over five nights, 10G per night.

This limit has no effect on a placement rule that matches directories only. The from fileset command determines whether files, directories, or both are matched. This command is described above.

For a placement rule without a schedule, this limit applies to the one-and-only run of the rule. If the original fileset exceeds this limit, the left-over files from that fileset remain on their source share(s) indefinitely. New files that belong in the fileset are created at the target share(s), before they have any size, so they are not blocked by this limit.

For example, the following command sequence sets a 50G limit on the "docs2das8" rule:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule docs2das8
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# limit-migrate 50g
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# ...
```

Removing the Limit

By default, a placement rule migrates until all matching files are removed from the source share(s). Use the no limit-migrate command to return to this default:

no limit-migrate

For example, the following command sequence removes any migration limit in the "mvTars" rule:

```
bstnA6k(gbl)# namespace archives
bstnA6k(gbl-ns[archives])# volume /home
bstnA6k(gbl-ns-vol[archives~/home])# place-rule mvTars
bstnA6k(gbl-ns-vol-plc[archives~/home~mvTars])# no limit-migrate
bstnA6k(gbl-ns-vol-plc[archives~/home~mvTars])# ...
```

Applying a Schedule (optional)

You can optionally run the placement rule on a schedule. The schedule determines when the rule scans the back-end filers behind the volume to find files for migration. This applies to existing files and/or directories that belong in your fileset; if there are a large number of files, you can use a schedule so that the migrations occur in stages and during off-peak hours. By default, the placement rule migrates all existing files on the first scheduled run; to spread the load out over time, you can use duration in the schedule or limit-migrate in the placement rule.

Whether or not you use a schedule, the rule places all *new* files as clients create them. By default, the rule also watches all client changes inline, and migrates any file that changes to match the source fileset. A schedule has no effect on new or newly-changed files.

"Creating a Schedule" on page 12-27 has full details on creating a schedule. To apply a schedule to the placement rule, use the schedule command in gbl-ns-vol-plc mode:

schedule *name*

where ***name*** (1-64 characters) identifies the schedule. Use the show policy schedule command to list all schedules.

For example, the following command sequence applies an hourly schedule to the "mvTars" rule:

```
bstnA6k(gbl)# namespace archives
bstnA6k(gbl-ns[archives])# volume /home
bstnA6k(gbl-ns-vol[archives~/home])# place-rule mvTars
bstnA6k(gbl-ns-vol-plc[archives~/home~mvTars])# schedule hourly
bstnA6k(gbl-ns-vol-plc[archives~/home~mvTars])# ...
```

Removing the Schedule

Use no schedule to use only a single, initial volume scan and migrate all files and/or directories found in that scan. This makes the placement rule migrate all existing files and/or directories in a single session:

no schedule

Again, a schedule (or lack of schedule) has no effect on new files, or files that clients change to match the fileset.

For example:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule docs2das8
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# no schedule
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# ...
```

Disabling Inline Notifications (optional)

Clients make changes to files that may cause them to be selected by a file-placement rule; for example, a client could rename a file or change its size. By default, the file-placement rule monitors all client changes *inline* and migrates any files that newly-match the source fileset. This occurs on an unscheduled basis. You have the option to wait for the next scheduled volume scan; from gbl-ns-vol-plc mode, use no inline-notify to disable the rule's inline notifications:

no inline-notify

This has no effect on *new* files or directories; the rule directs these so that they are created on the target share. No migrations occur in this case.

We recommend that you run this command only in a file-placement rule with a schedule (recall "Applying a Schedule (optional)" on page 14-16, above). Unless the rule runs on a schedule, this command disables all migrations after the rule's initial run.

For example, the following command sequence turns off inline notifications to the dailyArchive rule:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# no place-rule dailyArchive
bstnA6k(gbl-ns-vol-plc[medarcv~/rcrds~dailyArchive])# no inline-notify
bstnA6k(gbl-ns-vol-plc[medarcv~/rcrds~dailyArchive])# ...
```

Re-Enabling Inline Notifications

Inline notifications are recommended for rules without schedules. To reinstate inline notifications, use the inline-notify command:

inline-notify

For example, the following command sequence reinstates inline notifications for the emptyRH rule:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule emptyRH
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# inline-notify
```

```
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~emptyRH])# ...
```

Configuring Progress Reports

The next step in configuring a placement rule is optional but strongly recommended: setting up progress reports. Progress reports show all the milestones and results of a file-placement execution. The policy engine generates a report each time the schedule fires and invokes the rule.

Note

If the rule has no schedule, the rule generates a single report when it first runs.

This only generates a report if the rule executes some action, such as scanning a back-end filer or migrating at least one file or directory. It does not report on placement of new files, created by clients through the VIP.

By default, a placement rule generates no reports. From gbl-ns-vol-plc mode, use the report command to generate reports for the current rule:

report *prefix*

where ***prefix*** (1-64 characters) is the prefix to be used for the rule's reports. Each report has a unique name in the following format: *prefixYearMonthDayHourMinute*.rpt (for example, xrayBkup200403031200.rpt for a report with the "xrayBkup" prefix).

For example, the following command sequence enables reporting for the "dos2das8" rule:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule docs2das8
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# report docsPlc
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# ...
```

Generating Verbose Reports

Placement reports are terse by default. To make them verbose, use the optional verbose flag at the end of the report command:

report *prefix* verbose

where ***prefix*** is explained above.

For example, the following command resets “docs2das8” to produce verbose reports:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule docs2das8
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# report docsPlc verbose
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# ...
```

Deleting Empty Reports

By default, a placement rule creates a report every time it executes some action (such as scanning a back-end filer), even in cases where no files or directories are migrated. To remove any empty report files created this way, use the optional delete-empty flag at the end of the report command:

report *prefix* [verbose] delete-empty

where ***prefix*** and **[verbose]** are explained above.

For example, the following command resets “docs2das8” to delete any empty reports:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule docs2das8
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# report docsPlc verbose delete-empty
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# ...
```

Disabling Reports

From gbl-ns-vol-plc mode, use no report to stop generating placement reports for the current rule:

no report

For example, the following command sequence disables reporting for the rule, “mvTars:”

```
bstnA6k(gbl)# namespace archives
bstnA6k(gbl-ns[archives])# volume /home
bstnA6k(gbl-ns-vol[archives~/home])# place-rule mvTars
bstnA6k(gbl-ns-vol-plc[archives~/home~mvTars])# no report
bstnA6k(gbl-ns-vol-plc[archives~/home~mvTars])# ...
```

Enabling the Placement Rule

The final step in configuring any rule is to enable it. By default, the rule is disabled and ignored by policy software. Use the enable command to enable the rule.

enable

For example, the following command sequence enables the “docs2das8” rule:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule docs2das8
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# enable
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# ...
```

Tentatively Enabling the Rule

A tentatively-enabled rule is configured to appear in the system logs (syslog) as “tentative,” showing the potential effects of the rule if it was enabled. (The log component, POLICY_ACTION, creates the syslog messages; syslog access and log components are described in the *CLI Maintenance Guide*.) Tentative rules do not change policy enforcement, but they do consume processing time in the policy software. From gbl-ns-vol-plc mode, use the enable tentative command to tentatively enable the placement rule.

enable tentative

Use show logs syslog or grep *pattern* logs syslog to view the syslog.

For example, the following command sequence puts the “docs2das8” rule into a tentative state:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule docs2das8
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# enable tentative
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# ...
```

Disabling the Rule

Disabling the rule removes it from consideration. Use no enable from gbl-ns-vol-plc mode to disable a placement rule.

no enable

For example, the following command sequence disables the “docs2das8” rule:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# place-rule docs2das8
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# no enable
bstnA6k(gbl-ns-vol-plc[wwmed~/acct~docs2das8])# ...
```

Showing the Effects of File Placement

A file-placement rule moves files and/or directories to new shares. To show the effects of the file-placement rule, you can run an nsck report: nsck *namespace* metadata-only shows file/directory locations on back-end shares

You can invoke this command from priv-exec mode. It generates a report file that you can see with show reports. The metadata-only report is named “metadata_only.*id*.rpt” by default, where *id* identifies the nsck job. Use show, tail, or grep to view the contents of the report. See “Showing Metadata” on page 5-7 of the *CLI Maintenance Guide* for full details on all nsck report commands and their report formats.

Removing the Placement Rule

You can remove a placement rule to both disable it and delete its configuration. Many file-placement rules can manipulate directory mastership so that directory trees grow naturally on desired filers. If all directory masters are placed correctly, the managed volume creates new files and directories under them by default; the file-placement rule is no-longer needed.

Use the no form of the place-rule command to remove a placement rule:

no place-rule *name*

where ***name*** (1-64 characters) identifies the rule to be removed.

For example, the following command sequence removes the placement rule, “placeOnSAN19,” from the “medarcv~/rcrds” volume:

```
bstnA6k(gbl)# namespace medarcv
bstnA6k(gbl-ns[medarcv])# volume /rcrds
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# no place-rule placeOnSAN19
bstnA6k(gbl-ns-vol[medarcv~/rcrds])# ...
```

If the rule matches files and/or directories based on their names and/or ages instead of their paths, you may want to keep the rule indefinitely. The file-placement rule can continue to match against newly-created files/directories, steering them as needed. It can also re-assess existing files/directories as they age, or as clients change them, and migrate them as needed.

Changing Rule Order

The policy software enforces its rules in order. Whenever two rules conflict, the higher-order rule is enforced and the lower-order rule is not. Conflicts arise for files and/or directories that match two different rules; each rule may attempt to place the file or directory on a different target. By default, rules are ordered on a first-come-first-served basis; the first rule you enter is of the highest order, and the last rule is the lowest order.

You can change the rule order only for placement rules that use filesets as their sources. The priority cannot change for shadow-copy rules (which are always highest priority), placement rules that drain shares (which are second priority), and share farms (which are lowest priority). Fileset-placement rules are grouped together between the second and last priority groups; they are lower-priority than drain rules, but they take precedence over all share-farm rules.

Use the policy order-rule command in gbl-ns-vol mode to change the rule order for fileset-placement rules:

policy order-rule *rule1* {before | after} *rule2*

where

rule1 (1-64 characters) identifies a rule to move,

before | after is a required choice to set the new position for *rule1*, and

rule2 (1-64 characters) identifies a second rule, whose order stays the same.

Use the show policy *namespace* command to see the current rule order for a namespace. Refer back to "Focusing on One Namespace" on page 12-9.

If ***rule1*** already precedes ***rule2***, you cannot place it **before** rule2. Conversely, you cannot place ***rule1*** **after** ***rule2*** if it is already after it.

For example, the following command sequence shows the rule order in the "wwmed" namespace and then puts a file-placement rule, "placeTest," directly after the "docs2das8" rule:

```
bstnA6k(gbl)# show policy wwmed

Namespace:        wwmed

  Rule                                                          Status
Priority   Volume                Rule                 Vol. Scan    Migration
---------  --------------------  -------------------- ------------------------
    1      /acct                 placeTest            Complete      Complete
    2      /acct                 xls2archive          Complete      Complete
    3      /acct                 docs2das8            Complete      Complete
    4      /acct                 drain_share_rule_for_share_it5   Disabled      Disabled
    5      /acct                 fm1                  Complete      Complete
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
```

```
bstnA6k(gbl-ns[wwmed])# policy order-rule placeTest after docs2das8
bstnA6k(gbl-ns[wwmed])# ...
```

Moving the Rule to the Beginning or End

You can use the first or last keyword to move the file-placement rule to the beginning or end of the list:

policy order-rule *rule1* {first | last}

where **first | last** sets the new position for *rule1*.

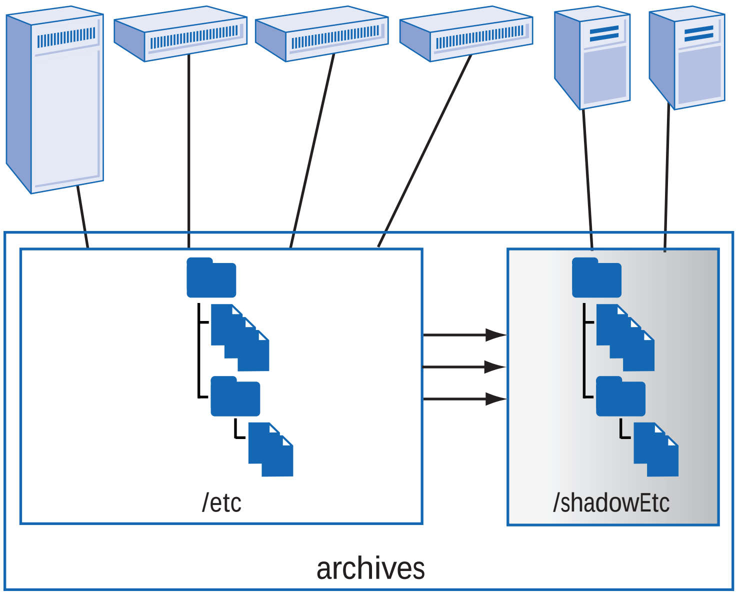
For example, the following command sequence moves the “docs2das8” rule to the first position:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns[wwmed])# policy order-rule docs2das8 first
bstnA6k(gbl-ns[wwmed])# ...
```

Chapter 15

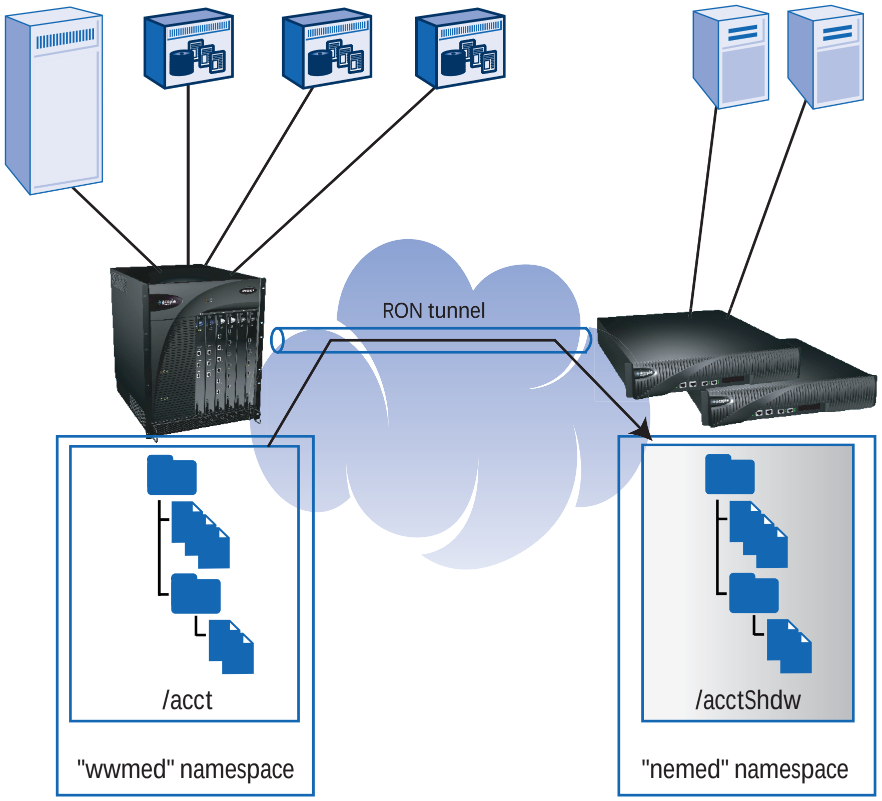
Shadowing a Volume

You can create a continuously-updated copy of a managed volume, called a *shadow volume*. The shadow volume can be on the same switch as its source volume (as shown below), or it can be hosted on another switch in the same RON (shown on the next page).



Shadow volumes have applications for implementing a Content-Delivery Network (CDN), backing up client data, or disaster recovery. For a CDN, you can configure several shadow volumes for the same source volume, each at a different switch. To back up volume files, you can funnel several source volumes into the same shadow volume. In case of disaster at the source volume, you can reverse the shadow volume's configuration and allow read-write access to the "shadowed" files.

The examples in this chapter configure a source volume on a single ARX®6000 and its shadow volume on a redundant pair. The redundant pair is two ARX®1000 switches:



The switch with the source volume is called the *source switch* and any switch with a shadow volume is called a *target switch*.

Before You Begin

Shadow volumes are commonly deployed on separate switches from the source volume, as pictured above. Before you configure the shadow volume on a target switch, you must first

1. make a RON tunnel from the source switch to the target switch (see Chapter 5, *Joining a RON*, in the *CLI Network-Management Guide*), and
2. add a namespace and source volume at the *source* switch (see Chapter 7, *Configuring a Namespace* and Chapter 9, *Adding a Managed Volume*).
3. add a namespace at the *target* switch.

You then perform the procedures in this chapter: you start by adding a shadow volume to the target switch, then you configure a shadow-copy rule at the source switch.

Adding a Shadow Volume (Target Switch)

The first step in shadowing a volume is configuring a second managed volume as a shadow. A shadow volume is different from a standard managed volume in that it can only contain replicas of source-volume files, and only the policy engine can write to it.

A shadow volume starts as managed volume (see “Adding a Volume” on page 7-21). To create the shadow volume on a different switch from the source volume, log into the target switch’s CLI or GUI and add a new volume. Once the volume is created and you are in gbl-ns-vol mode, use the shadow command to change the managed volume into a shadow volume:

```
shadow
```

The CLI presents a prompt to warn that all extraneous files will be removed from the shadow volume’s shares; you must answer **yes** for the volume to become a shadow volume. During the first shadow-copy operation, the rule replaces any files that are different from their source-file counterparts, and then it deletes all files that are *not* among the source files.

Choose a shadow volume with at least as much storage capacity as its source volume(s).

Note: The volume must be disabled when you change it into a shadow volume.

For example, the following command sequence creates a shadow volume to be used for the “wwmed~/acct” volume later. Note that this CLI session occurs on a different switch from the one where the “/acct” volume was created; this volume is created on “prtlndA1k,” a chassis in the same RON as the “bstnA6k” chassis. The namespace is also different; “nemed” instead of “wwmed.” The two volumes reside on separate switches and namespaces for added redundancy.

```
prtlndA1k(gbl)# namespace nemed
prtlndA1k(gbl-ns[nemed])# volume /acctShdw
This will create a new volume.

Create volume '/acctShdw'? [yes/no] yes
prtlndA1k(gbl-ns-vol[nemed~/acctShdw])# shadow

This will cause the all shares in the volume to be erased.

Are you sure you want to remove all data? [yes/no] yes
prtlndA1k(gbl-ns-vol[nemed~/acctShdw])#
```

Compatible Software Releases

The source and target switches can typically run different software releases, to facilitate staged software upgrades in you network. At press time, any combination of releases 2.1.4, 2.4, and 2.4.1 - 2.4.3 are compatible with this release.

Allowing Modifications

The shadow-copy rule will be modifying the metadata in the target volume, so the target volume must permit metadata modifications. This does not affect clients, who will have read-only access to the volume. It only applies to the shadow-copy rule itself. Use the modify command to enable file modifications by the rule. (The modify command was discussed in an earlier chapter; recall "Allowing the Volume to Modify on Import" on page 9-9.)

For example, the following command sequence permits modifications in the /acctShdw volume:

```
prtlndA1k(gbl)# namespace nemed
prtlndA1k(gbl-ns[nemed])# volume /acctShdw
prtlndA1k(gbl-ns-vol[nemed~/acctShdw])# modify

Automatically re-enable volume modify mode during NSCK rebuild? [yes/no] yes
prtlndA1k(gbl-ns-vol[nemed~/acctShdw])# ...
```

Adding a Share

A share in a shadow volume is configured the same way as any other managed-volume share, but it is used differently. The shadow-copy rule will replace all files on a shadow-volume share when it first runs; a share in a shadow volume is used exclusively for file replicas from the source volume. To guard against a single point of failure, we strongly recommend that you use shares from filers *other* than those used for the source volume. For maximum protection, use shares from filers at another ARX in the RON.

For example, the following command sequence adds two shares to the /acctShdw volume:

```
prtlndA1k(gbl)# namespace nemed
prtlndA1k(gbl-ns[nemed])# volume /acctShdw
prtlndA1k(gbl-ns-vol[nemed~/acctShdw])# share back1
prtlndA1k(gbl-ns-vol-shr[nemed~/acctShdw~back1])# filer das-p1 path /lhome/exports/BU
prtlndA1k(gbl-ns-vol-shr[nemed~/acctShdw~back1])# enable
prtlndA1k(gbl-ns-vol-shr[nemed~/acctShdw~back1])# exit
prtlndA1k(gbl-ns-vol[nemed~/acctShdw])# share back2
prtlndA1k(gbl-ns-vol-shr[nemed~/acctShdw~back2])# filer das-p2 path /export/home/bkup
prtlndA1k(gbl-ns-vol-shr[nemed~/acctShdw~back2])# enable
```

```
prtlndA1k(gbl-ns-vol-shr[nemed~/acctShdw~back2])# exit
prtlndA1k(gbl-ns-vol[nemed~/acctShdw])# ...
```

Turning Off Shadowing

You can turn off shadowing to convert a shadow volume back to a managed volume. This makes the volume ineligible for shadow copies and opens it up for client writes; all the shadow-copied files become fully accessible. From gbl-ns-vol mode, use the no shadow command to disable shadowing for the current volume:

no shadow

You must disable the shadow volume before you turn off shadowing.

A warning message appears, indicating that this erases any unpublished files and shadow databases. This means that files copied to the shadow volume to a hidden staging area, not yet "published" in their final directories, will be removed rather than moved into the final directory. Enter **yes** to continue.

For example, the following command sequence turns off shadowing for the "/buTest" volume:

```
bstnA6k(gbl)# namespace archives
bstnA6k(gbl-ns[archives])# volume /buTest
bstnA6k(gbl-ns-vol[archives~/buTest])# no enable
bstnA6k(gbl-ns-vol[archives~/buTest])# no shadow

This will cause any un-published files and shadow databases to
be erased.

Are you sure you want to proceed? [yes/no] yes
bstnA6k(gbl-ns-vol[archives~/buTest])# ...
```

Specifying a Fileset to Copy (Source Switch)

The next step in shadowing a volume is to choose a fileset to be “shadowed.” This occurs at the source volume, on the source switch. The fileset can include all files in the volume or a smaller set of files based on file names and/or file-access times. You can create complex filesets by intersecting two or more filesets (for example, choosing the *.mp3 files accessed in the last week) and/or joining them in a union (for example, choosing all .mp3 files *plus* all files of *all* types that were accessed in the last week). Refer back to Chapter 13, *Grouping Files into Filesets*, for a full range of options in creating filesets.

For example, the following command sequence occurs on “bstnA6k,” the switch with the source volume. These commands create an all-inclusive fileset called “worthSaving:”

```
bstnA6k(gbl)# policy-filename-fileset worthSaving
This will create a new policy object.

Create object 'worthSaving'? [yes/no] yes
bstnA6k(gbl-ns-vol-fs-name[worthSaving])# recurse
bstnA6k(gbl-ns-vol-fs-name[worthSaving])# exit
bstnA6k(gbl)# ...
```

Shadow Copies Are Not Cumulative

Note that if the fileset changes, the shadow changes with it. For example, consider an age-based fileset which changes based on client access. Files that are in the fileset one day may not be there the next. The shadow copy does not keep old files that are no longer in the source fileset: the changes to the shadow copy are not cumulative.

Configuring a Shadow-Copy Rule (Source Switch)

The final step in volume shadowing is to create a shadow-copy rule. A *shadow-copy rule* replicates the fileset in the source volume over to the shadow volume; if a file changes later, that file is replicated again. If a file is deleted, then its copy is also deleted. From gbl-ns-vol mode (in the *source* volume), use the shadow-copy-rule command to create a shadow-copy rule:

shadow-copy-rule *name*

where ***name*** (1-64 characters) is a name you choose for this rule.

The CLI prompts you for confirmation before creating the new rule; enter **yes** to continue. This places you into gbl-ns-vol-shdwcp mode, where you must identify the source fileset and the target shadow volume, make a schedule for the rule, then enable the rule. There are also several optional commands in this mode.

For example, the following command sequence creates a shadow-copy rule, “DRrule,” for the “/acct” volume. Like the previous example, this occurs on the “bstnA6k” switch:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRrule
This will create a new policy object.

Create object 'DRrule'? [yes/no] yes
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# ...
```

Identifying the Source Fileset

The next step in configuring a shadow-copy rule is to identify the source fileset. The shadow-copy rule keeps replicating all files from the source fileset onto the shadow volume. From gbl-ns-vol-shdwcp mode, use the from fileset command to specify a source fileset:

from fileset *fileset-name*

where ***fileset-name*** (1-64 characters) identifies the source fileset.

You can only use one source fileset. If you want to use additional filesets as sources, create a union fileset (see "Joining Filesets" on page 13-20). You can re-issue the from command to change from one source fileset to another.

For example, the following command set selects the "worthSaving" fileset as a source for the shadow copy:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRrule
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# from fileset worthSaving
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# ...
```

Choosing a Shadow-Volume Target

The next step in configuring a shadow-copy rule is to choose a target volume for the shadow copy. A target must be a shadow volume in the same RON. From gbl-ns-vol-shdwcp mode, use the target command to identify a shadow volume:

target [[hostname *host*] namespace *name*] volume *shadow*

where

hostname *host* (optional, 1-128 characters) identifies the target switch that hosts the shadow volume. If you omit this, the source switch is assumed. The switch must be on the same RON as the source switch; use the show ron command for a list of all switches on the RON (see "Showing the RON Configuration" on page 5-5 of the *CLI Network-Management Guide*).

namespace *name* (optional, 1-30 characters) identifies the namespace containing the shadow volume. If you omit this, it defaults to the current namespace.

shadow (1-256 characters) identifies the shadow volume.

For example, the following command sequence selects "/acctShdw" as a shadow volume for "DRrule:"

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRrule
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# show ron
```

```
Switch Name             HA Peer Switch                                 Uptime
Status                  UUID                                  Management Addr
-----------------------------------------------------------------------------
bstnA6k                 (None)                               0 days, 00:45:01
ONLINE                  7eafc74e-6fa9-11d8-9ed7-a9126cfbac40         10.1.1.7

provA5c                 (None)                               0 days, 01:45:03
ONLINE                  df3d1b6e-8459-11d9-8899-d2d1d3d64a34       10.1.38.19

prtlndA1k               prtlndA1kB                           0 days, 01:39:01
ONLINE                  9a6eb9ac-6c6d-11d8-9444-9e00f495ff7e       10.1.23.11

prtlndA1kB              prtlndA1k                            0 days, 01:44:42
ONLINE                  88babb94-8373-11d8-963c-8cd82b83827e       10.1.23.12

bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# target hostname prtlndA1k
namespace nemed volume /acctShdw
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# ...
```

Using a Path in the Shadow Volume

You may want to consolidate several source volumes into a single shadow volume, for ease of backups. Each source volume can store its files in a separate directory in the shadow volume. Use the optional path argument to create such a directory inside the volume:

target [[hostname *host*] namespace *name*] volume *shadow* path *path*

where

host, ***name***, and ***shadow*** are described above, and

path (optional, 1-1024 characters) is the path where the copies will be stored. This is a path from the root of the volume.

For example, the following command sequence selects two directories as shadow-copy targets:

```
prtlndA1k(gbl)# namespace testns
```

```
prtlndA1k(gbl-ns[testns])# volume /users
prtlndA1k(gbl-ns-vol[testns~/users])# shadow-copy-rule bkup
prtlndA1k(gbl-ns-vol-shdwcp[testns~/users~bkup])# target volume /shdw path /users
prtlndA1k(gbl-ns-vol-shdwcp[testns~/users~bkup])# exit
prtlndA1k(gbl-ns[testns])# volume /admin
prtlndA1k(gbl-ns-vol[testns~/admin])# shadow-copy-rule bkAdm
prtlndA1k(gbl-ns-vol-shdwcp[testns~/admin~bkAdm])# target volume /shdw path /admin
prtlndA1k(gbl-ns-vol-shdwcp[testns~/admin~bkAdm])# ...
```

Removing the Shadow-Volume Target

From gbl-ns-vol-shdwcp mode, use no target to remove a shadow-volume target:

no target [[hostname *host*] namespace *name*] volume *shadow* [path *path*]

where

host (1-128 characters) identifies the target switch that hosts the shadow volume. The default is the local (source) switch.

name (1-30 characters) identifies the namespace containing the shadow volume. The default is the current namespace.

shadow (1-256 characters) identifies the shadow volume.

path (optional, 1-1024 characters) is a specific directory to stop targeting.

If you remove the target, the shadow-copy rule is effectively disabled.

For example, the following command sequence removes a target volume and path from the “bkup” rule:

```
prtlndA1k(gbl)# namespace testns
prtlndA1k(gbl-ns[testns])# volume /users
prtlndA1k(gbl-ns-vol[testns~/users])# shadow-copy-rule bkup
prtlndA1k(gbl-ns-vol-shdwcp[testns~/users~bkup])# no target volume /shdw path /users
prtlndA1k(gbl-ns-vol-shdwcp[testns~/users~bkup])# ...
```

Applying a Schedule

A shadow-copy rule requires a schedule. Use the gbl schedule command to create one; refer back to "Creating a Schedule" on page 12-27 for details.

Note: You cannot use a schedule with a fixed duration (see "Setting the Duration (optional)" on page 12-28). If a duration was too short for the shadow copy to finish, the shadow copy would fail.

To apply a schedule, go to gbl-ns-vol-shdwcp mode and use the schedule command:

schedule *name*

where ***name*** (1-64 characters) identifies the schedule. Use the show policy schedule command to list all schedules.

For example, the following command sequence applies an hourly schedule to the shadow-copy rule, "DRrule:"

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRrule
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# schedule hourly
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# ...
```

Configuring Progress Reports

The next step in configuring a shadow-copy rule, optional but strongly recommended, is to arrange for progress reports. Progress reports show all the milestones and results of a shadow-copy execution. The policy engine generates a report each time the schedule fires and invokes the rule.

By default, no reports are generated. From gbl-ns-vol-shdwcp mode, use the report command to generate shadow-copy reports for the current rule:

report *prefix*

where ***prefix*** (1-64 characters) is the prefix to be used for the rule's reports. Each report has a unique name in the following format: *prefixYearMonthDayHourMinute*.rpt (for example, home_backup200403031200.rpt for a report with the "home_backup" prefix).

Use the show reports command for a list of all reports. Use show reports *report-name* to read a report, show reports status *report-name* for a one-line summary, grep to search through the report, or tail to tail a report as it is being written.

For example, the following command sequence enables reporting for the rule, "DRrule:"

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRrule
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# report DRetc
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# ...
```

Generating Verbose Reports

Shadow-copy reports are terse by default. To make them verbose, use the optional verbose flag at the end of the report command:

report *prefix* verbose

where ***prefix*** is explained above.

For example, the following command resets "DRrule" to produce verbose reports:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRrule
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# report DRetc verbose
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# ...
```

Including Identical Files in Reports

If files are identical on both the source and shadow volumes, the rule does not transfer the file. By default, identical files are omitted from the shadow-copy reports. Use the optional list-identical flag to include these files in the report:

report *prefix* [verbose] list-identical

where ***prefix*** and **[verbose]** are explained above.

For example, the following command sets the “insurDR” rule to produce verbose reports and include identical files:

```
bstnA6k(gbl)# namespace insur
bstnA6k(gbl-ns[insur])# volume /claims
bstnA6k(gbl-ns-vol[insur~/claims])# shadow-copy-rule insurDR
bstnA6k(gbl-ns-vol-shdwcp[insur~/claims~insurDR])# report insurDR verbose list-identical
bstnA6k(gbl-ns-vol-shdwcp[insur~/claims~insurDR])# ...
```

Deleting Empty Reports

By default, a shadow-copy rule creates a report every time it runs, even in cases where no files are copied between volumes. To remove any empty-report files created this way, use the optional delete-empty flag at the end of the report command:

report *prefix* [verbose] [list-identical] delete-empty

where ***prefix***, **[verbose]**, and **[list-identical]** are explained above.

For example, the following command resets “DRrule” to delete any empty shadow-copy reports:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRrule
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# report DRetc delete-empty
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# ...
```

Disabling Reports

From gbl-ns-vol-shdwcp mode, use no report to stop generating shadow-copy reports for the current rule:

no report

For example, the following command sequence disables reporting for the rule, "buHome:"

```
bstnA6k(gbl)# namespace archives
bstnA6k(gbl-ns[archives])# volume /home
bstnA6k(gbl-ns-vol[archives~/home])# shadow-copy-rule buHome
bstnA6k(gbl-ns-vol-shdwcp[archives~/home~buHome])# no report
bstnA6k(gbl-ns-vol-shdwcp[archives~/home~buHome])# ...
```

Supporting Local Groups (CIFS)

A Windows filer can support *Global Groups*, which are managed by Domain Controllers, and/or *Local Groups*, which are unique to the filer. Local groups have their own Security IDs (SIDs), unknown to any other Windows machine. When filers behind a source volume use local groups, they have SIDs that are unrecognized at the target volume's filers. This invalidates any Access Control Entries (ACEs) with these SIDs on the shadow volume; members of the local groups lose their privileges on the shadow volume. To resolve this problem, you must first prepare all of the target filers before you enable the shadow-copy rule:

- all local-group and local-user names must be configured on all filers behind *both* volumes, and
- all groups must contain the same users on those filers.

For example, if filers behind the source volume have a local group named "doctors," you must add a new "doctors" local group to all filers behind the target volume. The membership of these groups must match at all of the filers. This preparation is required so that all doctors can access the files at the source and shadow volumes.

This problem is similar for a volume with multiple CIFS filers behind it; recall "Supporting Filers with Local Groups" on page 9-18.

Caution

If the source and shadow volumes are in two different Windows domains, SID translation may require customized preparation. Contact Acopia Support before using this command for inter-domain shadow-copy rules.

Translating Local SIDs

After all local groups are duplicated on all source and destination filers, you must configure the shadow-copy rule to translate them. When SID translation is enabled, the rule finds a file's group name (such as "doctors") at the source volume, then looks up the SID for that group name at the destination filer. This introduces a slight performance penalty, but it ensures that doctors can access their files on both the source and shadow volumes. From gbl-ns-vol-shdwcp mode, use the sid-translation command to enable SID translations for the current shadow-copy rule:

sid-translation

For example, the following command sequence causes "DRrule" to translate all SIDs:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRrule
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# sid-translation
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# ...
```

Failing On SID-Translation Errors

If a local group at a source filer is not configured at the target, SID translations fail for that local group. By default, the shadow-copy rule copies the original SID (a binary number) to the shadow volume's filer. The Access Control Entry (ACE) with the SID does not function at the shadow volume, though it is preserved; if the file is copied back to the source volume later, the SID will still be valid there. This can be useful in a disaster-recovery application.

To prevent the rule from copying a file that fails its SID translation, use the fail-on-errors option at the end of the command:

sid-translation fail-on-errors

For example:

```
bstnA6k(gbl)# namespace insur
bstnA6k(gbl-ns[insur])# volume /claims
bstnA6k(gbl-ns-vol[insur~/claims])# shadow-copy-rule insurDR
bstnA6k(gbl-ns-vol-shdwcp[insur~/claims~insurDR])# sid-translation fail-on-errors
```

```
bstnA6k(gbl-ns-vol-shdwcp[insur~/claims~insurDR])# ...
```

Note: Some filer servers can be configured to return an error for an invalid SID (STATUS_INVALID_SID, STATUS_INVALID_OWNER, and/or STATUS_INVALID_PRIMARY_GROUP) but accept the file or directory anyway. You may want to discount these errors from these particular file servers. You can set this up for each errant file server from gbl-ns-vol-shr mode, using the sid-translation ignore-sid-errors command (recall "Ignoring SID Errors from the Filer (CIFS)" on page 9-36).

Disabling SID Translation

You can stop the shadow-copy rule from translating SIDs. This implies one of two scenarios:

- none of the filers behind the source volume use local groups, or
- clients do not require read access at the shadow volume.

This is the default for shadow-copy rules. Use no sid-translation to stop SID translation:

no sid-translation

For example, the following command sequence disables SID translation for the "insurDR" rule:

```
bstnA6k(gbl)# namespace insur
bstnA6k(gbl-ns[insur])# volume /claims
bstnA6k(gbl-ns-vol[insur~/claims])# shadow-copy-rule insurDR
bstnA6k(gbl-ns-vol-shdwcp[insur~/claims~insurDR])# no sid-translation
bstnA6k(gbl-ns-vol-shdwcp[insur~/claims~insurDR])# ...
```

Copying Files Opened through CIFS

By default, the shadow-copy rule opens each file for read-only access, thus blocking any other applications from writing to the file. This ensures the integrity of the copied file. However, the back-end filer does not allow the shadow-copy rule to open the file for read-only access if any other application already has the file open for writes, or for an impending deletion. The shadow-copy rule cannot copy these open files if they are left open for the duration of the shadow-copy run. Some common Microsoft applications, such as Microsoft Word, hold a file open for writes as long as the client application is working with the file. (Note that other applications, such as Notepad and WordPad, open the file only long enough to read it into memory; these applications rarely pose a problem for the shadow-copy operation.)

You can use the allow-shared-access command to open each file with all access types allowed: read, write, and delete. These access permissions make it possible to read a file even if another application is writing to it or deleting it at the same time. This introduces the risk of copying files between writes, so that the shadow-copy of the file is compromised. Use the allow-shared-access command only if this risk is acceptable:

allow-shared-access

For example, the following command sequence causes “DRrule” to allow unlimited access to files as they are copied:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRrule
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# allow-shared-access
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# ...
```

This makes it very likely that this shadow-copy rule, “DRrule,” will successfully copy any open file.

Allowing Only Reads During the Shadow Copy

By default, the shadow-copy rule only allows read access from other applications while it copies a file. This ensures that no other application can change or delete the file in the middle of the copy operation, but it also makes it more likely that the rule will fail to copy some previously-opened files. To return to this default, use the no allow-shared-access command:

no allow-shared-access

For example, the following command sequence causes the “insurDR” rule to block all writes and deletes while it copies files:

```
bstnA6k(gbl)# namespace insur
bstnA6k(gbl-ns[insur])# volume /claims
bstnA6k(gbl-ns-vol[insur~/claims])# shadow-copy-rule insurDR
bstnA6k(gbl-ns-vol-shdwcp[insur~/claims~insurDR])# no allow-shared-access
```

```
bstnA6k(gbl-ns-vol-shdwcp[insur~/claims~insurDR])# ...
```

This increases the chances of the rule failing to copy some open files; you can use two CLI commands to find and close all open files while the rule is running. The show cifs-service open-files command displays a list of all files that are held open by a client application (see "Listing Open Files in a CIFS Service" on page 9-9 of the *CLI Maintenance Guide*). To close one from the CLI, use close cifs file (see "Closing an Open File" on page 9-12 of the same manual).

Support for Multi-Protocol Volumes

The target volume must support all of the protocols in the source volume. That is, if the source volume supports only CIFS, the target volume must also be in a CIFS-only namespace. If the source volume supports CIFS and NFSv3, the target volume must support both of those protocols, too.

File-Attribute Translations

If the filers behind the source volume are from a different vendor than the filers behind the target volume, file attributes require some translation. (File attributes are the owner, group owner, permissions, timestamps, and other metadata associated with the file.) The issues and solutions for file-attribute replication are the same as those for file migrations in a multi-protocol volume. For a full discussion, refer back to "File-Attribute Migrations" on page 12-49.

Disabling Target Pruning (optional)

After the first shadow-copy run, the process *prunes* the directory tree on the target volume; that is, it removes all files and directories that are not on the source volume. For very large directory trees, the scanning for this can be very time-consuming. Some installations have a source volume and a shadow volume that they already know are identical; you can save time at those sites by disabling the prune-target feature. From gbl-ns-vol-shdwcp mode, use the no prune-target command to disable pruning:

no prune-target

This is only relevant before the first shadow copy. Pruning rarely occurs after the first run of the rule.

For example, the following command sequence disables pruning for the shadow-copy rule, DRrule:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRrule
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# no prune-target
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# ...
```

Enabling Pruning

Target-pruning is enabled by default, to ensure that the source and shadow volumes match after the first shadow copy. From gbl-ns-vol-shdwcp mode, use the prune-target command to re instate this default:

prune-target

For example, the following command sequence enables pruning of shadow volume target volumes for the 'bkup' rule.

```
prtlndA1k(gbl)# namespace testns
prtlndA1k(gbl-ns[testns])# volume /users
prtlndA1k(gbl-ns-vol[testns~/users])# shadow-copy-rule bkup
prtlndA1k(gbl-ns-vol-shdwcp[testns~/users~bkup])# prune-target
prtlndA1k(gbl-ns-vol-shdwcp[testns~/users~bkup])# ...
```

Copying Only Directories with Changed Files (optional)

By default, a shadow-copy rule copies all directories from the source volume, whether or not they have files from the source fileset. This can result in empty directories in the shadow volume. Use the directory-copy as-needed command to copy only the directories that contain at least one file:

directory-copy as-needed

For example, the following command sequence stops the "DRrule" from copying empty directories:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRrule
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# directory-copy as-needed
```

```
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# ...
```

Copying All Directories

Some applications require the presence of empty directories at specific paths; for those cases, you can use the directory-copy full command. This reverts to the default behavior:

directory-copy full

This copies the full directory tree from the source volume, even if the source fileset resides in a small number of directories. The directories without any matching files are copied to the shadow volume as empty directories.

For example, the following command sequence copies all directories for the "DRrule:"

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRrule
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# directory-copy full
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# ...
```

Publishing All Files as a Group (optional)

By default, the shadow-copy rule publishes all of its successfully-transferred files even if some of the file transfers fail. Some applications (such as a CAD application) require all files to be synchronized; a single missing file can cause these applications to fail. You can configure a shadow-copy rule so that no files are published in the shadow volume if any file fails to transfer properly.

To publish (or not publish) all files as a group, use the publish group command in gbl-ns-vol-shdwcp mode:

publish group

For example:

```
bstnA6k(gbl)# namespace ns3
bstnA6k(gbl-ns[ns3])# volume /cad1
bstnA6k(gbl-ns-vol[ns3~/cad1])# shadow-copy-rule cpRemote
bstnA6k(gbl-ns-vol-shdwcp[ns3~/cad1~cpRemote])# publish group
```

```
bstnA6k(gbl-ns-vol-shdwcp[ns3~/cad1~cpRemote])# ...
```

Publishing Individual Files

By default, a shadow-copy rule publishes all files that successfully transfer, whether or not some of the transfers fail. To return to this default, use the publish individual command:

publish individual

Note

This is generally recommended for CIFS volumes, since the shadow-copy rule does not transfer files opened by clients. See "Copying Files Opened through CIFS," above.

For example:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRrule
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# publish individual
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# ...
```

Setting a Bandwidth Limit

You can use the bandwidth-limit command to limit the amount of network bandwidth used by the shadow-copy rule. Use this command from gbl-ns-vol-shdwcp mode:

bandwidth-limit *rate*[K|M|G|T]

where

rate (100,000-4,000,000,000,000) is the allowable bandwidth for shadow-copy transfers, and

K|M|G|T (optional) is the units for the rate: Kbps (1,000 bps), Mbps (1,000,000 bps), and so on. This default is bits-per-second (BPS).

The bandwidth limit applies to all transfers by the current shadow-copy rule.

For example, this command sequence limits transfers by the rule, “DRrule,” to five million BPS:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRrule
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# bandwidth-limit 5M
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# ...
```

Removing any Bandwidth Limit

Use the no bandwidth-limit command to allow the rule to use unlimited bandwidth for its transfers:

no bandwidth-limit

For example, this command sequence removes any bandwidth limit from the “insurDR” rule:

```
bstnA6k(gbl)# namespace insur
bstnA6k(gbl-ns[insur])# volume /claims
bstnA6k(gbl-ns-vol[insur~/claims])# shadow-copy-rule insurDR
bstnA6k(gbl-ns-vol-shdwcp[insur~/claims~insurDR])# no bandwidth-limit
bstnA6k(gbl-ns-vol-shdwcp[insur~/claims~insurDR])# ...
```

Changing the Threshold for Delta Transfers (optional)

The shadow-copy rule generally copies the *delta* for each file that changed. That is, the source and shadow files are compared on a block-by-block basis, and only the blocks that changed are sent to the shadow volume. For large files, this saves network bandwidth. As files get smaller, the bandwidth savings can be offset by the cost of computation time.

By default, the shadow-copy rule performs delta transfers for files 100 MegaBytes or larger. Files smaller than 100 MegaBytes are transferred in their entirety. Note that both versions of the file (source and shadow) must exceed the delta threshold for the delta transfer to take place.

From gbl-ns-vol-shdwcp mode, use the delta-threshold command to set a different threshold:

delta-threshold *minimum-size*[k|M|G|T]

where

minimum-size (1-64) is the minimum size of a file that is eligible for delta transfer, and

k|M|G|T (optional) sets the size units: **k**ilobytes, **M**egabytes, **G**igabytes, or **T**erabytes. The default is bytes if you omit this. All of these values are 2-based, so a kilobyte is 1024 bytes, a megabytes is 1024*1024, and so on. There can be no spaces between the ***minimum-size*** and the unit letter; **20M** is correct, but **20 M** is invalid.

For example, the following command sequence sets a threshold of 500 megabytes for delta transfers:

```
bstnA6k(gbl)# namespace archives
bstnA6k(gbl-ns[archives])# volume /home
bstnA6k(gbl-ns-vol[archives~/home])# shadow-copy-rule buHome
bstnA6k(gbl-ns-vol-shdwcp[archives~/home~buHome])# delta-threshold 500M
bstnA6k(gbl-ns-vol-shdwcp[archives~/home~buHome])# ...
```

Reverting to the Default

To reset the threshold back to the default, 100 megabytes, use no delta-threshold:

no delta-threshold

For example:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRrule
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# no delta-threshold
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# ...
```

Enabling the Shadow-Copy Rule

The final step in configuring the shadow-copy rule is to enable it. By default, the rule is disabled and ignored by policy software. Use the enable command to enable the rule:

enable

For example, the following command sequence enables the rule, “DRrule:”

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRrule
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# enable
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRrule])# ...
```

Disabling the Rule

Disabling the rule stops all file replication. Use no enable from gbl-ns-vol-shdwcp mode to disable a shadow-copy rule:

no enable

For example, the following command sequence disables the “buHome” rule:

```
bstnA6k(gbl)# namespace archives
bstnA6k(gbl-ns[archives])# volume /home
bstnA6k(gbl-ns-vol[archives~/home])# shadow-copy-rule buHome
bstnA6k(gbl-ns-vol-shdwcp[archives~/home~buHome])# no enable
bstnA6k(gbl-ns-vol-shdwcp[archives~/home~buHome])# ...
```

Showing Shadow-Copy Status

Use the show shadow command to view the high-level status of the latest (or current) shadow-copy run:

show shadow

For every shadow-copy rule in every namespace, this shows the current progress of all shadow copies. Each rule has an overview section, a Target Status section listing all of the shadow volume targets, and two or more sections with more detail. The additional sections describe the most time-consuming parts of a shadow copy:

1. Copy Phase, where files are copied from the source volume to a staging area in the shadow volume.
2. Publishing Phase, where the files move from the staging area into their proper places in the directory tree.

For example, this shows two shadow-copy rules:

```
bstnA6k(gbl)# show shadow


Shadow Copy Status
==================


    Namespace                                        :   wwmed
    Source Volume                                    :   /acct
    Shadow Rule                                      :   DRrule
    Report File                                      :   DRetc_200701240413.rpt
    Fileset                                          :   worthSaving
    Shared Access Allowed                            :   Yes
    Bandwidth Limit                                  :   5.0 Mb/s (625.0 kB/s)

    ============================================================================
      Processing Started                             :   Jan 24 04:13
      Processing Completed                           :   Jan 24 04:25
      Elapsed Time                                   :   00:11:46
      Operating Mode                                 :   Full tree walk
      Tree Walk Reason                               :   Initial run
      Publishing Mode                                :   Individual
      Current Phase                                  :   Completed

      Target Information
      ------------------
        prtlndA1k: nemed:/acctShdw/                  :   Successful
      Copy Phase Information
      ----------------------
        Phase Started                               :   Jan 24 04:13
        Phase Completed                             :   Jan 24 04:24
        Elapsed Time                                :   00:11:31
        Average Transmission Rate                   :   3.4 Mb/s (433.9 kB/s)
        Total Files/Directories                     :   4,460
        Files/Directories Scanned                   :   4,460
        Files/Directories Skipped                   :   0
        Files/Directories Processed                 :   4,460
            Identical Files/Directories             :   0
            New Files/Directories                   :   4,460
            Updated Files/Directories               :   0
        Full Update Bytes Sent                      :   296,771,860 (283M)
        Delta Update Bytes Sent                     :   0 (0)
            Effective Data Bytes                    :   0 (0)

      Publishing Phase Information
      ----------------------------
        Phase Started                               :   Jan 24 04:24
        Phase Completed                             :   Jan 24 04:25
        Elapsed Time                                :   00:00:11
        Database Records Scanned                    :   4,460
        Files/Directories Published                 :   4,907
            New Files/Directories                   :   4,460
            Renamed Files/Directories               :   0
            Updated Files/Directories               :   447
            Removed Files/Directories               :   0


Shadow Copy Status
==================

    Namespace                                       :   insur
    Source Volume                                   :   /claims
    Shadow Rule                                          :   insurDR
    Report File                                          :   insurDR_200701240417.rpt
    Fileset                                              :   worthSaving

    ============================================================================
      Processing Started                                 :   Jan 24 04:17
      Processing Completed                               :   Jan 24 04:17
      Elapsed Time                                       :   00:00:07
      Operating Mode                                     :   Inline notification
      Publishing Mode                                    :   Individual
...

bstnA6k(gbl)# ...
```

Focusing on One Namespace

Add the namespace clause to specify one namespace:

show shadow namespace *name*

where ***name*** (1-30 characters) identifies a namespace with one or more source volumes.

For example:

```
bstnA6k(gbl)# show shadow namespace wwmed
...
```

Focusing on One Volume

After the optional namespace clause, you can add the volume clause to specify one source volume:

show shadow namespace *name* volume *vol-name*

where

name (1-30 characters) identifies the source namespace, and

vol-name (1-1024 characters) identifies the source volume.

For example:

```
bstnA6k(gbl)# show shadow namespace wwmed volume /acct
...
```

Focusing on One Rule

Add the Rule clause to specify one shadow-copy rule:

show shadow namespace *name* volume *vol-name* rule *rule-name*

where

name (1-30 characters) identifies the source namespace,

vol-name (1-1024 characters) identifies the source volume, and

rule-name (1-64 characters) is the name of the rule.

For example:

```
bstnA6k(gbl)# show shadow namespace wwmed volume /acct rule DRrule
...
```

Monitoring the Progress of a Shadow Copy

Use show reports to find the report for the shadow-copy rule, then use tail reports *report-name* follow to monitor the report as it is created.

Note

All shadow-copy reports are written to the ARX that hosts the *source* volume.

For example, the following command sequence finds a report on the "bstnA6k" chassis and tails it:

```
bstnA6k(gbl)# show reports


  reports
    Codes: At=Command Scheduler, Diag=Collect Diag-Info, Dstg=Destage,
           ExMp=Export Mapping, Imp=Import, Inc=Inconsistencies,
           MdO=Metadata Only, MdU=Metadata Upgrade, MgMd=Migrate Metadata,
           NIS=NIS Update, Plc=Place Rule, Rbld=Rebuild, Rm=Remove,
           RmNs=Remove Namespace, RmSh=Remove Share, RsD=Restore Data,
           SCp=Shadow Copy, Snapshot=Snapshot,
           SuEn=Enable Subshare Inconsistencies,
           SuIn=Export Subshare Inconsistencies, Sum=Summary,
           SuSh=Export Subshares, Sync=Sync Files/Dirs, SySh=Sync Shares
      adminSessions_200701240331.rpt Jan 24 03:31  1.4k        At   DONE: 7 in 00:00:00
      adminSessions_200701240446.rpt Jan 24 04:46  1.4k        At   DONE: 7 in 00:00:00
      auto-sync.8._claims.rpt Jan 24 03:59  2.9k        Sync DONE: 16 in 00:00:00
      cifsExportSubshares_200701240327.rpt Jan 24 03:27  1.1k        SuSh DONE: 3 in
00:00:00
      cifsExportSubshares_200701240330.rpt Jan 24 03:30  1.1k        SuSh DONE: 3 in
00:00:00
      DRetc_200701240413.rpt  Jan 24 04:25  2.3M        SCp  DONE: 4460 in 00:11:46
      cifs_only.rpt           Jan 24 03:46  3.7k        ExMp DONE: 7 in 00:00:00
      cifsExportSubshares_200701240332.rpt Jan 24 03:32  1.1k        SuSh DONE: 3 in
00:00:00
...

bstnA6k(gbl)# tail reports DRetc_200701240413.rpt follow
    /planner/findShare.fm                                              prtlndA1k:
nemed:/acctShdw/: Full update (190,712 bytes sent)
   /planner/findShare.fm                                 190,464  New(1)
    /planner/productOverview.fm                                        prtlndA1k:
nemed:/acctShdw/: Full update (194,808 bytes sent)
   /planner/productOverview.fm                           194,560  New(1)

    /planner/intro.fm                                                  prtlndA1k:
nemed:/acctShdw/: Full update (199,928 bytes sent)
   /planner/intro.fm                                     199,680  New(1)
    /planner/ron.fm                                                    prtlndA1k:
nemed:/acctShdw/: Full update (214,264 bytes sent)
   /planner/ron.fm                                       214,016  New(1)
    /planner/ndmp.fm                                                   prtlndA1k:
nemed:/acctShdw/: Full update (204,024 bytes sent)
   /planner/ndmp.fm                                      203,776  New(1)
    /planner/globalServer.fm                                           prtlndA1k:
nemed:/acctShdw/: Full update (223,480 bytes sent)
   /planner/globalServer.fm                              223,232  New(1)
    /planner/showRunning.fm                                            prtlndA1k:
nemed:/acctShdw/: Full update (210,168 bytes sent)
```

```
   /planner/showRunning.fm                             209,920  New(1)
    /planner/filesetPolicy.fm                                            prtlndA1k:
nemed:/acctShdw/: Full update (221,432 bytes sent)
   /planner/filesetPolicy.fm                           221,184  New(1)
    /planner/securityMgmtSvcs.fm                                         prtlndA1k:
nemed:/acctShdw/: Full update (225,528 bytes sent)
   /planner/securityMgmtSvcs.fm                         225,280  New(1)
    /planner/cliOperatorIX.fm                                            prtlndA1k:
nemed:/acctShdw/: Full update (285,944 bytes sent)
...
```

You can also use show reports *report-name* to read a report, or grep to find a string in the report.

Report Format

The shadow-copy report shows the progress of tree walks and replications (the “Copy Phase” in the show shadow output) on a file-by-file basis. The standard output from show shadow (discussed above) appears at the end of this report.

For example,

```
bstnA6k(gbl)# show reports DRetc_200701240413.rpt
**** Shadow Copy Report: Started at Wed Jan 24 04:13:21 2007 ****
**** Software Version: 2.05.000.09925 (Jan 23 2007 17:40:03) [nbuilds]
**** Hardware Platform: ARX-6000

Shadow Copy Status
==================

    Namespace                                     :   wwmed
    Source Volume                                 :   wwmed:/acct
    Shadow Rule                                   :   DRrule
    Fileset                                       :   worthSaving
    Shared Access Allowed                         :   Yes
    Bandwidth Limit                               :   5.0 Mb/s (625.0 kB/s)

    ============================================================================
      Processing Started                          :   Jan 24 04:13
      Operating Mode                              :   Full tree walk
      Tree Walk Reason                            :   Initial run
      Publishing Mode                             :   Individual
      Target Information
      ------------------
        prtlndA1k: nemed:/acctShdw/

Shadow Copy File Details
========================

   Filename                                                               Size
Status
    --------------------------------------------------------------------------------
--------------  ------------------------------
   /layer3.fm.lck
prtlndA1k: nemed:/acctShdw/: Full update (212 bytes sent)
   /layer3.fm.lck
84  New(1)
   /rework_vpn.fm.lck
prtlndA1k: nemed:/acctShdw/: Full update (212 bytes sent)
   /rework_vpn.fm.lck
84  New(1)
   /shellCmds_winXP.fm.lck
prtlndA1k: nemed:/acctShdw/: Full update (212 bytes sent)
   /shellCmds_winXP.fm.lck
84  New(1)
   /relNotes.arx.2.1.4.pdf
prtlndA1k: nemed:/acctShdw/: Full update (88 bytes sent)
   /relNotes.arx.2.1.4.pdf
45  New(1)
   /planner
prtlndA1k: nemed:/acctShdw/: Full update (88 bytes sent)
   /planner
8,192  New(1)
   /sampleNet.html
prtlndA1k: nemed:/acctShdw/: Full update (1,024 bytes sent)
/: Full update (88 bytes sent); Reason: FileMissing
   /workarea.2.1.4
22  New(1)
...


Shadow Copy Status
==================
    Namespace                                   :   wwmed
    Source Volume                               :   wwmed:/acct
    Shadow Rule                                 :   DRrule
    Fileset                                     :   worthSaving
    Shared Access Allowed                       :   Yes
    Bandwidth Limit                             :   5.0 Mb/s (625.0 kB/s)

    ============================================================================
      Processing Started                        :   Jan 24 04:11
      Processing Completed                      :   Jan 24 04:23
      Elapsed Time                              :   00:12:41
      Operating Mode                            :   Full tree walk
      Tree Walk Reason                          :   Initial run
      Publishing Mode                           :   Individual
      Current Phase                             :   Completed

      Target Information
      ------------------
        prtlndA1k: nemed:/acctShdw/             :   Successful

      Copy Phase Information
      ----------------------
        Phase Started                           :   Jan 24 04:11
        Phase Completed                         :   Jan 24 04:23
        Elapsed Time                            :   00:12:39
        Average Transmission Rate               :   3.1 Mb/s (395.8 kB/s)
        Total Files/Directories                 :   4,460
        Files/Directories Scanned               :   4,460
        Files/Directories Skipped               :   0
        Files/Directories Processed             :   4,460
            Identical Files/Directories         :   0
            New Files/Directories               :   4,460
            Updated Files/Directories           :   0
        Full Update Bytes Sent                  :   296,771,860 (283M)
        Delta Update Bytes Sent                 :   0 (0)
            Effective Data Bytes                :   0 (0)

      Publishing Phase Information
      ----------------------------
        Phase Started                           :   Jan 24 04:23
        Phase Completed                         :   Jan 24 04:23
```

```
        Elapsed Time                              :   00:00:05
        Database Records Scanned                  :   4,460
        Files/Directories Published               :   4,907
            New Files/Directories                 :   4,460
            Renamed Files/Directories             :   0
            Updated Files/Directories             :   447
            Removed Files/Directories             :   0

Total processed:           4,460
Elapsed time:           00:12:47
**** Shadow Copy Report: DONE at Thu Jan 24 04:23:55 2007 ****
```

Reformatting the Report

You can use the copy reports command to duplicate the report in a different format, XML or CSV (Comma-Separated Values). The duplicate's file extension indicates the desired format: to convert to XML, use a .xml extension; to convert to CSV, use .csv. The copy command is available in priv-exec mode:

copy reports *source destination*[.xml|.csv]

where

source (1-255 characters) specifies report to copy,

destination (1-255 characters) is the name of the duplicate report, and

[.xml|.csv] chooses the duplicate's format (XML or CSV, respectively).

A .txt or .rpt extension keeps the destination report in plain-text format.

For example, the following command sequence exits to priv-exec mode and creates an XML copy of a shadow report:

```
bstnA6k(gbl)# exit
bstnA6k# copy reports DRetc200403230239.rpt DRetc_3-23.xml
bstnA6k# ...
```

Truncating the Report

To conserve CPU cycles and/or internal-disk space, you may want to stop a shadow-copy report before it is finished. An oversized, CPU-intensive report could possibly have an effect on namespace performance. From priv-exec mode, use the truncate-report command to stop all report processing and truncate the report file:

truncate-report *name*

where ***name*** (1-255 characters) specifies report to truncate.

This only stops the policy engine from writing to the report; the shadow-copy processing continues.

For example, the following command finds a running shadow-copy report and truncates it:

```
bstnA6k(gbl)# show reports


reports
    Codes: At=Command Scheduler, Diag=Collect Diag-Info, Dstg=Destage,
           ExMp=Export Mapping, Imp=Import, Inc=Inconsistencies,
           MdO=Metadata Only, MdU=Metadata Upgrade, MgMd=Migrate Metadata,
           NIS=NIS Update, Plc=Place Rule, Rbld=Rebuild, Rm=Remove,
           RmNs=Remove Namespace, RmSh=Remove Share, RsD=Restore Data,
           SCp=Shadow Copy, Snapshot=Snapshot,
           SuEn=Enable Subshare Inconsistencies,
           SuIn=Export Subshare Inconsistencies, Sum=Summary,
           SuSh=Export Subshares, Sync=Sync Files/Dirs, SySh=Sync Shares
...
      DRetc_200701240413.rpt  Jan 24 04:25  2.3M        SCp  DONE: 4460 in 00:11:46
...

bstnA6k(gbl)# end
bstnA6k# truncate-report DRetc_200701240413.rpt
bstnA6k# ...
```

Copying the Source Volume to Multiple Targets

To copy a source volume to more than one target, create a separate shadow-copy rule for each target. Each shadow-copy rule can operate on its own schedule, or they can all use the same schedule. For example, the following command sequence sets up a second shadow-copy rule for the “wwmed~/acct” volume, which goes to a different target switch using the same schedule:

```
bstnA6k(gbl)# namespace wwmed
bstnA6k(gbl-ns[wwmed])# volume /acct
bstnA6k(gbl-ns-vol[wwmed~/acct])# shadow-copy-rule DRbkup
This will create a new policy object.

Create object 'DRbkup'? [yes/no] yes
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRbkup])# from fileset worthSaving
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRbkup])# target hostname provA5c
namespace wwmedBkup volume /acctBkup
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRbkup])# schedule hourly
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRbkup])# report DRbkup
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRbkup])# enable
bstnA6k(gbl-ns-vol-shdwcp[wwmed~/acct~DRbkup])# ...
```

Removing the Shadow-Copy Rule

You can remove a shadow-copy rule to both disable it and delete its configuration. Use the no form of the shadow-copy-rule command to remove a shadow-copy rule:

no shadow-copy-rule *name*

where ***name*** (1-64 characters) identifies the rule to be removed.

For example, the following command sequence removes the shadow-copy rule, “buHome,” from the “/home” volume:

```
bstnA6k(gbl)# namespace archives
bstnA6k(gbl-ns[archives])# volume /home
bstnA6k(gbl-ns-vol[archives~/home])# no shadow-copy-rule buHome
bstnA6k(gbl-ns-vol[archives~/home])# ...
```

Copyrights

Index